TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: B2646 – Informační technologie

Studijní obor: 1802R007 – Informační technologie

Bakalářská práce

Srovnání virtualizačních technologií

Comparison of virtualization technologies

Autor: **Petr Milichovský**

Vedoucí práce: **Mgr. Milan Keršláger**

V Liberci 10. 5. 2012

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Datum:

Podpis:

Poděkování

Touto cestu bych chtěl poděkovat především vedoucímu své bakalářské práce Mgr. Milanovi Keršlágerovi za jeho podporu a mnoho cenných rad při vytváření bakalářské práce.

Zde bych ještě chtěl poděkovat své rodině za podporu při studiu a vytvoření potřebného zázemí. Rodina mi byla vždy velkou oporou.

Abstrakt

Tato bakalářská práce se zabývá srovnáním virtualizačních technologií na softwaru VirtualBox, VMware a XEN. Pro srovnání virtualizačních technologií byla zvolena testovací sada Phoronix-test-suite a vlastní testovací nástroj. Nejdůležitějším hardwarovým komponentám, které jsou procesor, operační paměť, grafický adaptér a pevný disk byly vybrány jednotlivé testy ze sady Phoronix-test-suite. Testy byly vybrány takovým způsobem, aby byly jednoduché, ale na druhé straně poskytly relevantní data pro porovnání. V bakalářské práci jsou diskutovány a použity různé virtualizační technologie, které nám použité nástroje nabízejí. Dále byly navrhnuty doporučení pro používání jednotlivých nástrojů v různých oblastech. Výsledky byly poté porovnány a vyhodnoceny.

Klíčová slova

Virtualizace,

Paravirtualizace,

Úplná virtualizace,

Virtuální stroj,

Hypervizor,

VirtualBox,

VMware,

XEN,

Phoronix-test-suite

Abstract

This thesis deals with the comparison of virtualization technologies on software VirtualBox, VMware and XEN. A test suite Phoronix-Test-suite and own testing tool were selected for the comparison of virtualization technologies. The most important hardware components, which are CPU, RAM, video card and hard disk, were tested on the basis of selected Phoronix-test-suite tests. The tests were selected in such a way as to be easy, but on the other hand to provide relevant data for comparison. In the thesis, different virtualization technologies that the tools offer are discussed and used. Further, recommendations for the use of individual tools in different areas were designed. The results were then compared and evaluated.

Keywords

Virtualization,

Paravirtualization,

Full virtualization,

Virtual machine,

Hypervisor,

VirtualBox,

VMware,

XEN,

Phoronix-Test-suite

Obsah

Seznam obrázků

Seznam grafů

Seznam tabulek

Seznam zkratek

IT = Information technology – Informační technologie
MMU = Memory management unit – Jednotka správy paměti
RAM = Random access memory – Paměť s přímým přístupem
KiB = KiloByte – Kilo bajt
VA = Virtual adress – Virtuální adresa
LA = Linear adress – Lineární adresa
GPA = Guest physical address – Pseudo-fyzický adresní prostor
SPA = System physical address – Fyzický adresní prostor
TLB = Translation lookaside buffer
PIO = Programmed Input Output – Programovaný vstup výstup
DMA = Direct Memory Acess – Přímý přístup do paměti
IBM = International Business Machines – Firma
GPL = General public licence – Obecná veřejná licence
MIPS = Million Instructions Per Second – Milion instrukcí za vteřinu
MB/s = Megabyte per second – Megabajty za sekundu
MFLOPS = Floating-point Operations per Second - počet operací v plovoucí řádové čárce za sekundu
RPS = Requests per second – Požadavek za sekundu
UPS = Unit per second – Jednotka za sekundu
FPS = Frames per second – Snímky za sekundu

1 Úvod

Stále častěji se setkáváme téměř ve všech odvětvích informatiky, podnikání a průmyslu s úspornými opatřeními, které vedou ke snižování finančních nákladů. Každá větší, ale i menší firma vlastnící IT systémy řeší opakovaně v 5-10letých intervalech nákup nového hardwaru pro své zaměstnance. Virtualizace je technika umožňující, v případě že je navrhnuta a prováděna správným způsobem, ušetřit nemalé finanční prostředky. Myšlenka virtualizace je taková, že na jednom „superpočítači“ vytvoříme potřebný počet virtuálních strojů neboli hostů. Každého hosta pak přidělíme uživatelům ve firmě a nemusíme se starat o hardware jednotlivých osobních počítačů. Odpadá tak i přirozeně údržba o velké množství osobních počítačů. Staráme se jen o jeden superpočítač. Což je pochopitelně jednodušší z hlediska údržby, komfortnější a profesionálnější řešení. Toto ovšem není jediný způsob užití virtualizace. Hojně využívána je také v oblasti serverové techniky. Provozovat několik serverových služeb pomocí virtualizace na jednom stroji zajišťuje nejen finanční úsporu při nákupu hardwaru, ale i robustnější řešení provozovaných služeb. Nejedná se jen o ušetřené peníze z nákupu hardwaru. Virtualizované stroje mohou být lépe přizpůsobeny potřebám uživatelů, snáze se používají, případně můžeme uživateli odepřít nepotřebné funkce a aplikace systému. Virtualizace nenabízí jen celé systémy, virtualizovaná mohou být jednotlivě hardwarové komponenty. Virtualizace všeobecně nabízí mnoho výhod oproti klasickému řešení.

Cílem práce bylo porovnat mezi sebou virtualizační technologie. Pro tento účel byly vybrány tři virtualizační nástroje, a to VirtualBox, VMware a XEN. Jako testovací nástroj byla zvolena open source sada Phoronix-test-suite. Srovnávat technologie lze mnoho různými způsoby. V této bakalářské práci byl zvolen způsob testování na principu zátěžových testů jednotlivých hardwarových komponent virtualizovaného počítače. Toto řešení nám dovoluje porovnat způsoby virtualizace na téměř všech dnes používaných operačních systémech.

2 Teoretická část

Virtualizace

Virtualizace v informatice nejčastěji znamená postup nebo techniku, která uživateli umožňuje přistupovat k hardwarovým nebo aplikačním prostředkům jiným způsobem, než kdyby fyzicky existovali. V praxi to znamená možnost virtualizace celého počítače, popřípadě jen specifického softwarového prostředí a aplikací. Uživateli to přináší mnohem jednodušší přizpůsobení jeho potřebám bez jakýchkoliv změn hardwaru nebo operačního systému. Virtualizace nabízí široké spektrum možností a úrovní jak virtualizovat. Uživatel si může vybrat od virtualizace aplikací přes virtualizaci jádra operačního systému až po virtualizaci celé počítačové architektury. Podnětem pro nasazování virtualizace jsou nejčastěji důvody pro zvýšení efektivity využívání systémových zdrojů, dosažení větší robustnosti a dostupnosti systémů.[1] [9]

Virtuální stroj

V informatice je pojmem virtuální stroj označován software vytvářející virtuální prostředí mezi operačním systémem a hardwarem počítače. Pomocí tohoto může uživatel provozovat software na abstraktním stroji. Označení virtuální stroj však nese častěji obraz počítače neboli virtuální počítač. Uživatel používáním tohoto obrazu není nijak omezen, aplikace se chovají stejně jako na fyzickém stroji. Tyto aplikace se spouštějí a pracují na virtuálním stroji, kde nemohou ovlivnit chod programů na fyzickém stroji. Na jednom fyzickém stroji lze spustit více těchto obrazů a dosáhnout tak provozu více uživatelů zároveň. Uživatelé jsou od sebe odděleni a prakticky jeden o druhém neví.[1] [9]

2.1 Systémová virtualizace

Tento pojem souhrnně označuje v informatice pojmy: plná virtualizace, paravirtualizace atd. V dalších kapitolách se budeme věnovat těmto pojmům podrobněji. Nyní představíme pojem systémová virtualizace. Jedná se o virtualizaci, která je spuštěna na fyzickém stroji ten nazýváme hostitelský systém. Hostitelský systém rozdělujeme na více virtuálních strojů. Někdy jsou tyto systémy nazývány také jako obrazy. Jestliže jsme si zvolili cestu plné virtualizace. Potom jako virtuálním strojem zde rozumíme celé výpočetní prostředí, důsledně virtualizujeme všechny části

počítače. Do virtuálního stroje pak zpravidla instalujeme operační systém, který můžeme bez omezení provozovat. V případě paravirtualizace musíme před spuštěním provést drobné úpravy. V praxi se používá systémová virtualizace obvykle z několika důvodů. Nejdůležitějšími důvody, proč se virtualizace začala tak masivně používat, jsou náklady spojené s nákupem a následným provozem výpočetní techniky a výpočetní výkon. Virtualizací se snažíme využít tyto výkonové prostředky co možná nejlépe. Virtuální stroje řídí virtualizační manažer neboli hypervizor, který má na starost veškerou správu potřebnou pro běh vizualizovaného systému. Stará se o nezávislost každého jednotlivého obrazu. Více o hypervizoru je popsáno v samostatné podkapitole.[3] [1]

Konsolidace

Konsolidace v překladu znamená upevnění, ustálení a urovnání. Co si pod tímto pojmem představit z hlediska virtualizace bude vysvětleno v této podkapitole. Hlavní myšlenkou je překonvertovat stávající celé fyzické systémy do jednotného virtuálního prostředí a snížit tím potřebné množství fyzických strojů potřebné pro chod systémů. Toto je bezesporu hlavní a nejdůležitější důvod pro zavádění virtualizace. Ve světě bez virtualizace obyčejně provozujeme na každém stroji pouze jednu službu. Co se týče výpočetního výkonu fyzických strojů je nutno si uvědomit, že virtualizované stroje nemusí dosahovat výkonu fyzických strojů. Virtualizované systémy mohou mít nižší výkon než fyzické stroje. V praxi dokonce ve většině případů mají nižší výkon. Je to dáno tím, že průměrné vytížení se pohybuje okolo 20 %, zbylých 80 % výkonu často potřebujeme pro pokrytí zátěžových špiček. V konsolidovaném prostředí mohou systémy izolovaně provozovat více služeb. Je velmi pravděpodobné, že jejich špičky se v čase rozprostřou. Z tohoto důvodu si můžeme výpočetní výkon virtualizovaných strojů snížit oproti fyzickým strojům. Tato řešení vedou k rovnoměrnějšímu vytížení fyzických strojů, na kterých běží virtualizace. A také ke snížení nákladů spojených s nákupem hardwaru a jeho následného provozu.[3]

Migrace

Systémová virtualizace dovoluje přemístit, neboli migrovat virtuální stroj z jednoho fyzického počítače na druhý beze ztráty dat a stavových informací uložených v paměti. Migrace odstraňuje závislost programového vybavení na fyzickém vybavení

počítače neboli hardwaru. Představme si situaci, kde všechny fyzické stroje mají stejnou architekturu a používají stejný hypervizor. V takovéto situaci máme možnost zastavit virtualizovaný systém a zkopírovat jeho disk s pamětí na jiný systém. Po zkopírování můžeme systém obnovit a pokračovat v běhu. Pokud navíc všechny servery sdílí centrální úložiště dat se souborovým systémem podporujícím paralelní přístup z více uzlů, můžeme přemístění systému provádět v časech desetin vteřin. Tato technika se nazývá migrace za běhu. Dovoluje nám provádět údržbu systému, migrování systémů v případě výpadku nebo při zátěžových špičkách.[3]

Robustnost

Tento termín ve světě informatiky používáme, když chceme o některém systému říci, že je schopen přežít kritické situace. Jedná se o zhodnocení situace reálného prostředí, ve kterém bude systém umístěn. U složitějších systémů, které uchovávají nebo pracují s důležitými daty, je vyžadována jistá míra redundance pro zajištění robustnosti systému. V praxi nabývá tento požadavek ještě větší důležitosti. Je to zapříčiněno tím, že na jednom stroji obyčejně běží více než jedna služba. Pokud dojde k výpadku, přestanou fungovat všechny služby. Aby nedocházelo v nevirtualizovaném prostředí po výpadku systému ke ztrátě dat, můžeme obnovit původní stav nebo alespoň navrátit se do bezpečného stavu a zajistit tak konzistenci dat. Aby se toto dalo provést, aplikace musí zvládnout ukládat veškerá svá data a stavové informace na sdílené úložiště. Jestliže aplikace nedokáží ukládat svá data, potom musíme ke každému stroji pořídit záložní server a tyto dva stroje neustále synchronizovat. Toto je ale velice finančně náročné. Svět virtualizace je v tomto ohledu daleko jednodušší. Následná obnova po výpadku není řešena na úrovni aplikace, ale na úrovni celých systémů.[3]

Izolace

Systémová virtualizace nabízí vzájemnou izolaci mezi klienty. Každý klient disponuje vlastním virtuálním systémem včetně administrátorských práv. Klient si tak spravuje vlastní virtuální prostředí a může si ho libovolně konfigurovat, instalovat aplikace či například sdílet svá data. Vše je možné bez rizika.[3]

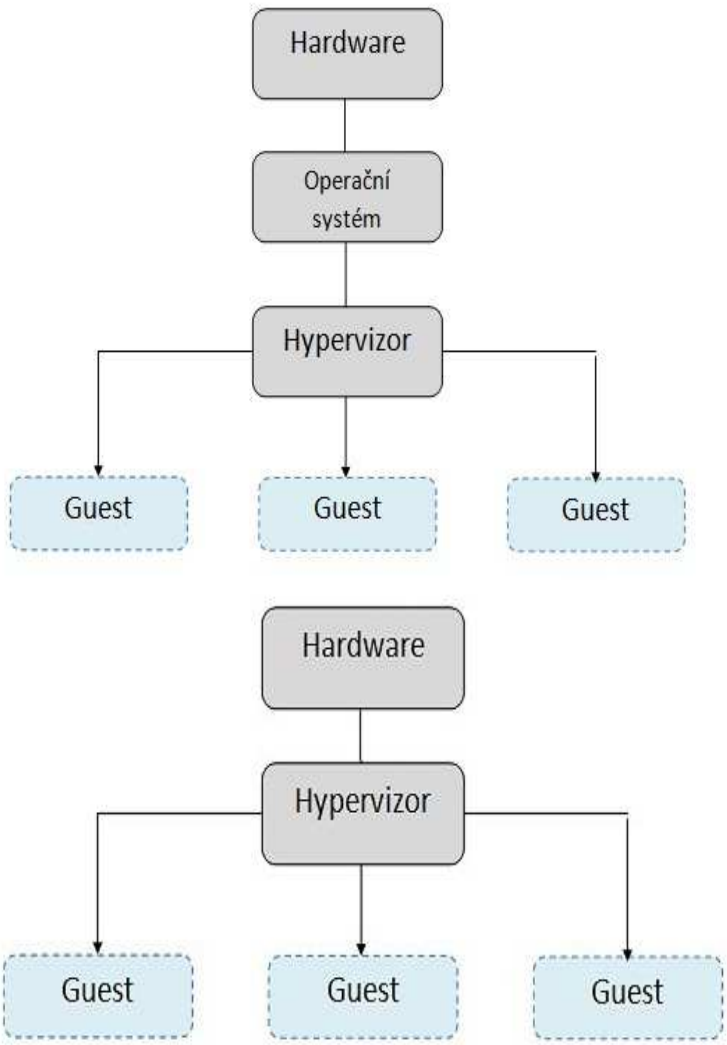
Vývoj

Velké uplatnění virtualizace najdeme také v odvětví testování a vývoje hardwaru, softwaru atd. Při vývoji nového systému, je nutné ho před uvedením do provozu nejprve otestovat. To vyžaduje například otestování jeho běhu na řadě operačních systémů v rozdílných konfiguracích. Virtualizace umožňuje vytvářet rozsáhlé databáze jakýchkoli systémů, tím vzniknou testovací prostředí. Systém dokáže pomocí dat uložených v databázi systém kdykoliv zavést. Následně můžeme testovat. Hojně využívanou vlastností virtualizace je také ladění operačního systému za běhu. Jelikož má hypervizor plnou kontrolu nad virtuálním systémem, může v případě vzniku systémové chyby lépe zjistit její příčiny. Poslední zde zmíněnou vlastností využívanou při testování je snapshot. Systém může za běhu kdykoliv uložit aktuální stav virtuálního systému a poté se k němu vrátit. Desktopové systémy tuto vlastnost obdobně používají při přechodu do hibernace.[3]

2.2 Hypervizor

Hypervizor neboli Virtuální strojový monitor je nejvyšší entitou, která řídí přístup virtuálních strojů k hardwaru počítače. Název hypervizor vyplývá z jeho nadřazenosti vůči hostitelskému počítači neboli supervizoru. Hypervizor dovoluje vytváření nových virtuálních počítačů a mazání těch, které jsou již nepotřebné. Hypervizor provádí plánování, přidělování a účtování zdrojů. Dále má na starost vzájemné oddělení virtuálních počítačů, aby nedocházelo ke vzájemnému sdílení dat a jejich poškození. Zprostředkovává bázové funkce operačního systému a v případě potřeby povoluje přístup k nepovoleným rutinám. Jeho funkce a chod nesmí být nijak ovlivňovány virtualizovanými stroji. V situaci, při níž by virtuální stroj nějakým způsobem poškodil chod hypervizoru, mohlo by dojít k poškození dat, či pádu hostovaného systému. Toto bezpečnostní opatření bývá u současných hypervizorů již ošetřeno a zřídka dochází k těmto chybám. Rozlišujeme dva druhy hypervizorů, nativní a hostovaný. Nativní hypervizor běží přímo na hardwaru fyzického stroje. Zde monitoruje a řídí běh virtualizovaných strojů z hlediska hardwarových prostředků. Vrstva hostovaného systému funguje pod hypervizorem. Hostovaný hypervizor je spouštěn v klasickém konvenčním prostředí operačního systému, není tak blízko hardwaru jako v případě nativního hypervizoru. Proto u tohoto řešení dochází k větší režii řízení hostovaných systémů. Toto rozdělení nám dává na výběr dvě možnosti

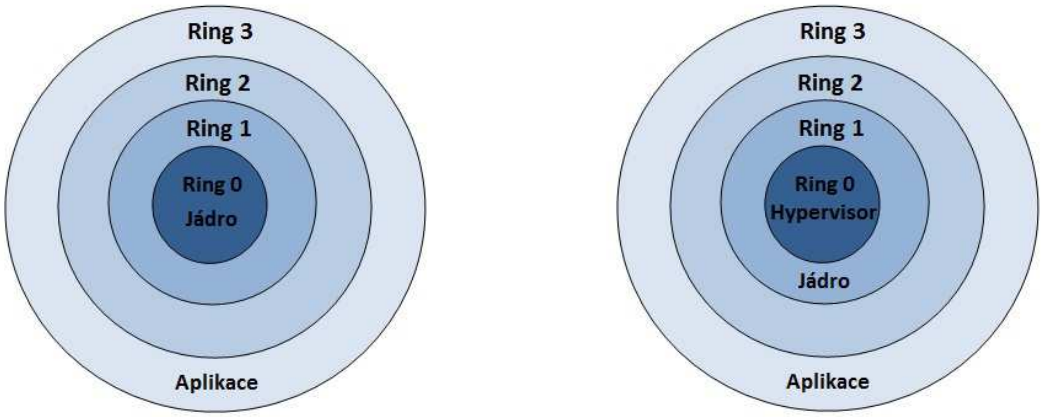
virtualizace a to hardwarovou a softwarovou. Hardwarová virtualizace disponuje vyšším výkonem, ale vyžaduje podporu v procesoru. Softwarová virtualizace má naopak větší režii, tudíž i menší výkon. Je to způsobeno do určité míry i tím, že jednoduché operace jsou obsluhovány komplexními obslužnými rutinami.[3] [9]



Obr. 1. Blokové schéma hostovaného a nativního hypervizoru

2.2.1 Virtualizace procesoru

Virtualizace procesoru je první problém, který je potřeba vyřešit. Procesor pracuje zpravidla alespoň ve dvou od sebe odlišných režimech: privilegovaném a neprivilegovaném. V privilegovaném režimu je dovoleno provádět privilegované strojové instrukce, které mohou nějakým způsobem ohrozit chod počítače. Jedná se o kontrolovaný přístup uživatele k hardware. Druhý režim se nazývá – neprivilegovaný neboli uživatelský. Jak název napovídá, zde běží všechny uživatelské programy. Jsou to programy, které nemohou narušit chod počítače. Bez použití virtualizace, například u procesorů Intel, běží operační systém počítače na nejvyšším režimu oprávnění, což je ring 0. Ostatní uživatelské programy jsou spuštěny s nejnižším stupněm ochrany na režimu ring 3. Při použití virtualizace běží hypervizor nejčastěji na nejvyšším stupni ochrany a jádro operačního systému je přesunuto do režimu ring 1. Tento postup se nazývá ring deprivileging Na Obr. 2 je znázorněno schéma režimů procesoru. Obrázek v levé části představuje řešení bez virtualizace, obrázek v pravé části znázorňuje situaci při použití virtualizace. Jednotlivé kruhy představují režimy procesoru. Od 0 do 3, stupnice znamená od nejvyššího stupně oprávnění po nejmenší stupeň oprávnění. Ostatní režimy se běžně nevyužívají.[3]



Obr. 2. Klasické rozdělení režimů procesoru a rozdělení při virtualizaci

2.2.2 Virtualizace MMU

Virtuální paměť je v informační technice způsob správy operační paměti počítače. Dovoluje přidělit běžícímu procesu adresní prostor paměti, který je větší nežli připojená fyzická paměť RAM. Adresní prostory lze sdílet, a efektivněji tak vytvářet procesy. Pro převod mezi virtuální a fyzickou adresou je nutná hardwarová podpora v procesoru nebo samostatný obvod. Procesor pak nadále rozlišuje mezi virtuálními a fyzickými adresami. Převod obstarává správa virtuální paměti. Každý proces vyžaduje po spuštění přidělení fyzických zdrojů. Přidělování paměti probíhá tak, že proces po spuštění obdrží od operačního systému oblast pro uložení programu a data. V případě, kdy proces potřebuje více operační paměti, požádá operační systém o další přidělení. Pokud systém nemá potřebné množství paměti, kolik je po něm vyžadováno, dojde k odsunutí části obsahu operační paměti na pevný disk. Následně systém provede úpravy ve stránkovacím registru procesu. Rozlišujeme dva základní modely implementace virtuální paměti, stránkování a segmentace.[3] [14]

Stránkování

Metoda správy paměti, kde strojové instrukce procesoru pracují s logickými adresami, které jednotka MMU převádí na fyzické adresy. Tak vzniká virtuální adresní prostor začínající od nuly pro všechny procesy. Při použití modelu stránkování je paměť rozdělena na větší úseky stejné velikosti, nazývané stránky. Velikost stránek je typicky 4 KiB. Virtuální adresní prostor obsahuje stejně velké stránky, které odpovídají stejně velkým stránkám ve fyzické paměti. Tato akce je využívána k minimalizaci počtu potřebných informací pro převod logických adres na fyzické adresy. Logické stránky svojí strukturou vytvářejí souvislý lineární prostor, avšak umístění fyzických stránek je po převodu zcela náhodné. Běžící proces, který dostal přidělenou virtuální operační paměť, má k dispozici souvislý adresní prostor. Ve skutečnosti jsou fyzické stránky fragmentovány. Pro proces je fragmentace neviditelná a není nutné ji řešit na rozdíl od segmentace.[3]

Segmentace

Metoda správy paměti, kde je procesu vytvořen virtuální adresní prostor začínající od nuly. Tímto odpadá potřeba relokace použitého strojového kódu. Strojové instrukce používají pro adresaci logické adresy neboli offsety. Fyzická adresa se získává součtem segment registru a offsetu. Model segmentace rozděluje paměť na různě veliké segmenty. Tyto segmenty mají délku uzpůsobenou opravdové potřebě procesu. Pro vytvoření virtuální adresy potřebuje segmentace dva speciální registry: segment a offset. Jejich součet pak představuje fyzickou adresu. Segment neboli paměťový prostor má přidělen každý běžící proces. Offset je ukazatel na začátek segmentu. S offsetem pak pracují strojové instrukce udržující si seznam odkazů na začátky segmentů. Segment nemá předem danou velikost. Velikost je určena až po porovnání dvou hodnot: limit a adres, které požadujeme – offset. Segmentový registr je procesu nepřípustný, jeho nastavení má na starost operační systém nebo hypervizor v privilegovaném režimu. Všechny procesy mají obyčejně k dispozici více segmentů odpovídajících logickému členění adresního prostoru procesu. Může být například rozdělen na datový, programový a zásobníkový segment.[3]

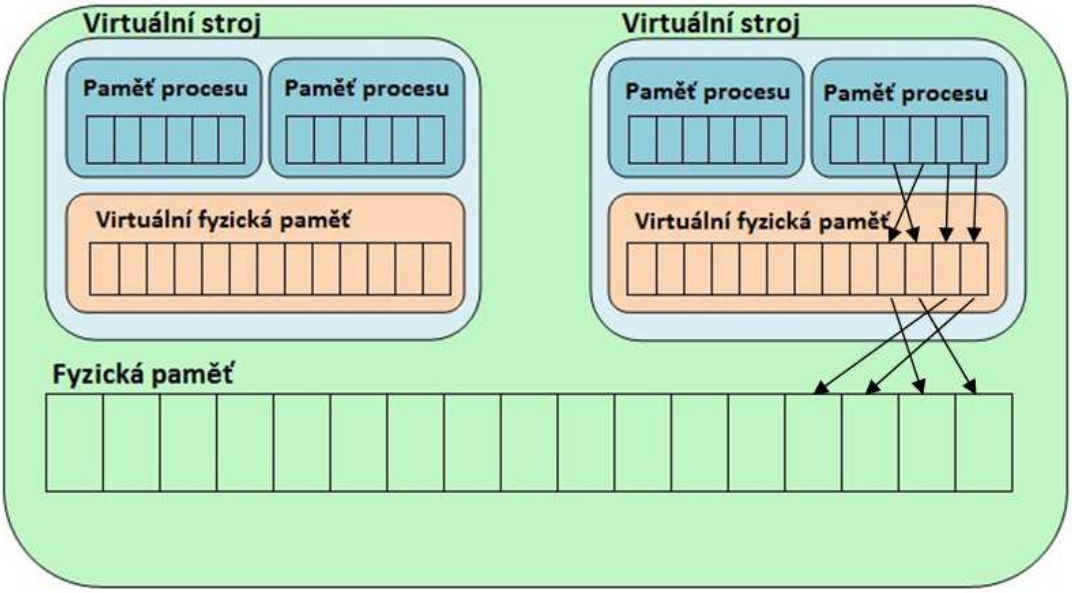
Adresní prostory a překlad adres

Nyní nahlédneme na operační paměť z hlediska systémové virtualizace. V této fázi potřebujeme vyřešit dva problémy. Prvním problémem je, jak správně dávkovat fyzickou paměť mezi virtualizované systémy. Dále je potřeba vyřešit záležitost jakým způsobem provádět překlad adres z virtualizovaného stroje na fyzické adresy.

- virtuální adresní prostor (virtual address VA) je prostor alokován pro proces běžící na virtuálním počítači
- lineární adresní prostor (linear address LA), překladem virtuální adresy procesu získáme lineární adresu, všechny procesy disponují vlastním lineárním prostorem
- pseudo-fyzický adresní prostor (guest physical address GPA), vzniká přeložením lineární adresy, jedná se o fyzické adresy virtuálního počítače
- fyzický adresní prostor (system physical address SPA), vzniká pomocí stránkováním na úrovni hypervizoru, je to skutečná fyzická adresa

Přidělování paměti je vyřešeno jádrem operačního systému. Jádro operačního systému, každému procesu zajistí vlastní virtuální adresový prostor a za pomoci překladových tabulek spravuje jejich zobrazení do fyzické paměti. Virtuální adresy používají aplikace ve virtualizovaných systémech. Vlastní překlad virtuálních adres na fyzické adresy má na starost komponenta MMU. Překlad můžeme provést již zmíněnými technikami, segmentací nebo stránkováním. Existuje i varianta kombinující obě tyto techniky. Základní model překládání adres vypadá následovně: jednotka MMU přeloží virtuální adresu pomocí tabulek segmentů na lineární adresu, která se následně tabulkou stránek přeloží na fyzickou adresu. Paměť procesu je složena ze stránek s logickými adresami, které jsou očíslovány jako stránky ve fyzické paměti. K převodu čísel logických stránek na čísla fyzických stránek je využívána speciální tabulka. Nazývá se tabulka stránek.

Tabulka stránek obsahuje údaje o procesech a jejich oprávněních, zda je stránka načtena v operační paměti, kdy byl jaký záznam naposledy změněn a tak dále. Protože převod stránek je velmi častá operace a tabulka stránek je uložena v operační paměti, je do procesoru vložena speciální cache paměť urychlující čtení z tabulky stránek. Cache paměť se nazývá Translation Lookaside Buffer (TLB). TLB obsahuje záznamy ze stránkovacích tabulek, které byly v poslední době použity. Při opakovaném přístupu na stejné adresy se již nemusí záznamy vyhledávat znovu.[3] [4] [5]



Obr. 3. Přidělování paměti ve virtuálním stroji [5]

2.2.3 Virtualizace I/O zařízení

Důležitá oblast, bez které by nebyla virtualizace plnohodnotná, je virtualizace vstupních a výstupních zařízení. Vzhledem k tomu, že zařízení, která můžeme k počítači připojit existuje nesčetné množství, jedná se o nejsložitější oblast. Každé zařízení má teoreticky jinou techniku zapojení a pracuje s konkrétními unikátními ovladači, jež se musejí do operačního systému nainstalovat. Instalace není triviální operací, proto virtualizace všech ovladačů vyžaduje velké úsilí. Prostředky pro běh a správu zařízení se snažíme zajistit na nižší úrovni, nejlépe na hardwaru počítače.[3]

I/O Operace

- **PIO (Programmed Input Output)**

  Jedná se o způsob přenosu dat po sběrnici mezi I/O zařízeními a operační pamětí. Přenos zajišťuje procesor. Procesor je touto operací plně zaměstnán, tudíž nemůže provádět jiné operace. Po obdržení zdrojových a cílových adres přenosu od operačního systému, začne procesor kopírovat data ze zdroje do svých registrů a poté do cílového umístění přenosu.[3]

- **DMA (Direct Memory Access)**

  Opačný způsob PIO přenosu dat mezi operační pamětí a I/O zařízeními. Data nepřesunujeme skrze procesor, ale přenos dat iniciuje samotné zařízení. Procesor již není plně zaměstnaný přenosem dat a může zpracovávat jiné operace. Procesor pouze vydá svolení k přenosu dat a nastaví důležité parametry k přenosu. O další přenos se již nestará. Zařízení vyčká na vhodnou dobu a provede přenos. Následně dá vědět procesoru vnějším přerušení o dokončení přenosu.[3] [7]

- **Vnější přerušení**

  Jedná se o asynchronní obsluhu I/O zařízení. [3] Jakmile procesor dostane požadavek k přerušení, přeruší vykonávání instrukcí, které měl vykonat a po dokončení rozpracované instrukce vykoná obsluhu vnějšího přerušení. Jakmile dojde k dokončení obsluhy přerušení, procesor se navrátí do stavu před přerušením a dále pokračuje v činnosti. Vnější přerušení se nazývá takto proto,

že přichází z vnějších zařízení. Vnější zařízení jsou například pevné disky nebo optické mechaniky. Zařízení mají možnost požádat procesor o vykonání instrukce, které potřebují bez toho, aby zjišťovaly, zdali je procesor zaneprázdněn.[6] Přerušení zprostředkovává procesoru řadič přerušení, speciální obvod přidělující přerušením priority, rozdělovat mezi procesory a další operace.

Typy virtualizace I/O zařízení:

- **Plná virtualizace I/O**

  Hypervizor virtualizuje celá zařízení. Emulované zařízení si zachovávají všechny své vlastnosti, nastavení a vzhled. Zařízení lze spouštět i na odlišných architekturách, než pro která byla napsána. Vstupní a výstupní zařízení nemusí skutečně existovat, můžeme virtualizovat stará zařízení. Virtualizovaný systém převede každou I/O operaci do nízkoúrovňových příkazů, které jsou srozumitelné pro emulovaná zařízení. Příkazy se předají hypervizoru. Hypervizor provede nutné úpravy a poté je přepošle k fyzickému zařízení. Emulovaná zařízení jsou zpravidla jiného typu nežli fyzická zařízení, hypervizor musí ještě zařídit konverzi příkazů mezi různými komunikačními protokoly. Nevýhodou tohoto řešení je vysoká správa a s tím spojený nízký výkon.[3]

- **Paravirtualizace I/O**

  Řešení softwarového rozhraní přináší výrazné navýšení výkonu, než dosahovala emulace. Virtualizované ovladače přistupují vědomě k virtuálním zařízením v hypervizoru, nikoli k hardwaru. Paravirtualizovaný systém využívá rozumněji existence hypervizoru než v případě plné virtualizace. Systém předá hypervizoru I/O příkazy v abstraktní podobě. A konverzi, kterou v úplné virtualizaci zajišťoval virtualizovaný systém, také přenechá hypervizoru. Celková složitost průběhu zpracování příkazů a požadavků se velmi sníží.[3]

- **Přímé mapování**

  Jednotlivým virtualizovaným systémům je přiřazen přímý přístup k I/O zařízením. Virtualizované systémy používají originální ovladače zařízení. Díky hardwarové podpoře nabízí přímé mapování téměř reálný výkon fyzického stroje. Účast hypervizoru při vykonávání I/O operací na tomto modelu virtualizace je minimální nebo nulová.[3]

- **Sdílení I/O zařízení**

  Metoda přímého mapování rozšířená o možnost sdílení zařízení mezi více virtualizovanými systémy. Jedná se o techniku efektivního sdílení PCI express zařízení. Zařízení musí být schopno obsloužit více souběžných dotazů od různých zařízení.[3]

2.3 Úplná virtualizace

Neboli také nazývána jako Emulace hardwaru. Jedná se o jednu z nejstarších virtualizačních technik. Tento způsob virtualizace se objevil v roce 1966 u systému nazvaného CP-40 a později u jeho následovníka, systému CP-67. Tyto systémy pocházejí z dílny IBM. Jednalo se o první počítač, kde si mohl uživatel spustit neboli virtualizovat jiný plnohodnotný počítač. Uživatel zde spouštěl jednoduchý jednouživatelský systém CMS. Aby bylo možné úplnou virtualizaci provozovat, je nutné, aby architektura hostovaného operačního sytému byla shodná s architekturou hostujícího operačního systému. Proto nelze vystavět úplnou virtualizaci na jakékoliv architektuře. V případě, kdy architektury hostujícího a hostovaného systému budou shodné, nemáme ještě úplně vyhráno. Existují jistá pravidla virtualizace, která shrnuli Gerald John Popek a Robert P. Goldberg. Jestliže chceme virtualizovat správně, měli bychom tyto podmínky dodržet.[3]

Ekvivalence

Operační systém nebo jiný software běžící pod hypervizorem by měl vykazovat totožný chod se softwarem, který běží přímo na hardwaru. V podstatě to znamená, že virtualizace vytvoří pro každý virtuální stroj přesnou kopii fyzické vrstvy počítače, tak aby měl každý obraz dostatečné prostředky pro chod a správu systému nebo softwaru. Za dostatečně ekvivalentní nemůžeme zpravidla pokládat rychlost a dostatek zdrojů potřebných pro běh. Jednotlivé virtuální stroje se musí dělit o procesorový čas. Znamená to, že každý obraz má ve skutečnosti k dispozici jen část skutečného procesorového času. U systémových zdrojů je tento problém obdobný.[3]

Bezpečnost

Hypervizor musí mít plnou kontrolu nad všemi virtualizovanými stroji, jak již bylo zmíněno v kapitole věnované hypervizoru, funkce a chod hypervizoru nesmí být ovlivňována virtualizovanými stroji. Virtualizované stroje musí být od sebe navzájem izolovány. Hypervizor přidělí všem systémům omezené množství prostředků, se kterými mohou pracovat. Veškeré operace nad systémovými zdroji obstarává hypervizor.[3]

Efektivita

Naprostá většina instrukcí by měla být prováděna přímo na hardwaru počítače. Aby byla virtualizace efektivní, je vyžadována poslední podmínka. Instrukce, které nemohou ovlivnit chod ostatních systémů, se snažíme provádět nativně.[3]

2.3.1 Virtualizace procesoru

K provozu úplné virtualizace je potřeba, aby architektura hostujícího systému obsahovala určité vlastnosti. Tyto vlastnosti nejsou u všech procesorů běžné. Pro chod úplné virtualizace tyto vlastnosti od architektury hostujícího systému vyžadujeme. Jestliže by architektura virtualizace zmíněné vlastnosti neobsahovala, nedošlo by k naplnění Popek-Goldbergových podmínek:

- Procesor podporuje alespoň dva režimy ochrany.

- Architektura podporuje asynchronní přerušení, skrze něhož může být procesor upozorněn na vzniklé události.

- Správa paměti podporuje dynamickou relokaci a bezpečnostní mechanismy pro práci s virtuální pamětí, například: stránkování a segmentace.

- Instrukce, které mají možnost změnit nastavení systémových zdrojů. Jako jsou například systémové registry, tabulky stránek, I/O zařízení. A instrukce, jejichž průběh nebo výsledek je ovlivněn touto konfigurací, spadají do podmnožiny privilegovaných instrukcí.

Poslední vlastnost, kterou po architektuře požadujeme, je stěžejní, protože ne všechny architektury ji implementují. Pokud ji implementují, tak pouze jen částečně. Jestliže všechny instrukce, které mají možnost změnit konfiguraci systémových zdrojů jsou privilegované, zavolání instrukce z neprivilegovaného režimu způsobí vnitřní přerušení. Následně se řízení a chod instrukce předá hypervizoru. Virtualizovaný systém nemá možnost ovlivnit konfiguraci fyzických zdrojů počítače, aniž by o tom hypervizor věděl. Bohužel řada dnešních procesorových architektur nemá řádně oddělené privilegované a neprivilegované instrukce. Zavoláním těchto nesourodých instrukcí v neprivilegovaném režimu nemusí vyvolat vnitřní přerušení a virtualizovaný systém se tak může dozvědět o nastavení fyzických zdrojů. Tyto instrukce, které mají přístup k nastavení fyzických zdrojů, nazýváme: kritické instrukce. Pro architektury obsahující kritické instrukce nelze vytvořit hypervizor, který by pracoval čistě na principu úplné virtualizace. Kritické instrukce se musí buď ošetřit softwarově, nebo navrhnout modifikaci architektury zohledňující virtualizaci.[3] [6]

2.3.2 Virtualizace MMU

Virtualizace MMU je založena na vytváření stínových kopií prostředků virtualizovaného systému. Hypervizor udržuje pro každý obraz virtualizovaného systému kopii tabulky stránek, takzvanou stínovou tabulku stránek. Jeho úkolem je zajistit konzistenci mezi stínovou a primární tabulkou stránek. Konzistence lze zajistit dvěma způsoby. První způsob by se dal nazvat: odepření přístupu zápisu do primární tabulky stránek. Hypervizor nepovolí virtualizovanému systému zapisovat do stránek paměti, kde je uložena primární tabulka. Vždy, když se pokusí virtualizovaný systém provést změnu této tabulky, dojde k vnitřnímu přerušení a řízení převezme hypervizor. Hypervizor pak provede požadovanou změnu a upraví stínovou tabulku. Druhý způsob se nazývá: líná synchronizace. Virtualizovaný systém má možnost samovolně měnit svoji primární tabulku a synchronizace se začíná řešit až po přístupu k změněnému záznamu. Zmíněné techniky generují velký počet vnitřních přerušení. Primární tabulka je důležitá pro správný běh algoritmů, správy paměti, synchronizace accessed a dirty bitů. Převážně se jedná o rozhodnutí, zdali zapsat stránky na disk a uvolnit tak místo v paměti.[3]

2.3.3 Virtualizace I/O zařízení

Technika, kterou úplná virtualizace ve většině případů používá pro virtualizaci I/O zařízení, je plná virtualizace. Tato technika implementuje kompletně běh hardwaru. Navenek se virtualizovaným systémům tváří virtualizované zařízení jako identické kopie fyzických zařízení. To umožňuje používat originální ovladače. Úplná virtualizace je velice náročná z hlediska spotřeby procesorového času a z hlediska implementace. Potřebujeme virtualizovat rozsáhlou skupinu zařízení a jejich časem přidávané nové funkce. Náročnost virtualizace obyčejně vede k implementaci pouze základních vlastností zařízení. Rozšíření a nové funkce často nebývají přístupné. Naopak výhodou úplné virtualizace I/O zařízení je jednoduché zavedení migrací virtualizovaných systémů. Seznam zařízení je celý uložen v paměti, proto ho lze jednoduše migrovat.[3]

2.4 Paravirtualizace

Další z výčtu virtualizačních technik, paravirtualizace. Jedná se o techniku, kde nedochází k plné emulaci hardwaru. Paravirtualizace předává virtuálním strojům virtualizovanou architekturu, která je velmi podobná té, na které se hodlá systém spouštět. Je to abstrakce reálného prostředí. V mnoha ohledech je lépe optimalizována a zjednodušena pro sdílení systémových prostředků. Některé možnosti procesoru mohou být omezeny a systém může rozpoznat, že běží ve virtuálním stroji. Hlavní myšlenka paravirtualizace je vyhnout se složitým problémům, které jsou spojeny s úplnou virtualizací. Paravirtualizovaná architektura nedisponuje žádnými kritickými operacemi. Namísto toho virtuální stroj obsahuje prostředky pro volání hypervizoru. V případě úplné virtualizace nebylo možné dosáhnout plného výkonu virtualizovaného systému. Tento způsob řešení nabízí mnohem více výpočetního výkonu. Systém totiž může plně využívat systémové prostředky bez vnější správy a režie. Odpadá i emulace celého hardwaru. Abychom mohli správně virtualizovat, potřebujeme docílit spolupráce mezi virtuálním strojem a hypervizorem, což znamená modifikovat virtualizovaný operační systém. Při zavádění paravirtualizace je tento úkon jeden ze stěžejních. Většina dnešních operačních systémů je připravena pro běh na různých platformách, tudíž jejich modifikace je snadná. Narazíme však u proprietárních operačních systémů, kde neznáme jejich zdrojové kódy. Tyto překážky nás nutí použít jen některé principy paravirtualizace. Často používaným modelem je plná virtualizace procesoru a pamětí

doplněná o paravirtualizaci vstupních a výstupních zařízení. Toto řešení nevyžaduje přílišné změny v operačním systému a zároveň nabízí výrazné zlepšení výkonu.[3]

2.4.1 Paravirtualizace procesoru

Pokud použijeme paravirtualizaci, pak hypervizor převezme nejvyšší stupeň ochrany a operační systém se přesune na režim ring 1. Aplikace běží stále na režimu ring 3. Touto migrací nebyla narušena hierarchie a operační systém je stále nad aplikačními programy, na druhou stranu operační systém již nemůže provádět privilegované instrukce. Pokud však bude procesor ve virtuálním stroji potřebovat provést privilegovanou instrukci, na kterou nemá oprávnění, pak dojde k přerušení a řízení převezme hypervizor, jenž operaci zkontroluje a provede ji tak, aby správně změnila stav virtuálního počítače. V této chvíli může virtuální počítač poznat, že běží ve virtuálním prostředí. Běží-li v režimu ring 1, má možnost číst některé části paměti, které jsou ve virtuálním počítači jiné než na fyzickém stroji. Pro paravirtualizaci je nutné upravit některé součásti operačního systému. Změny jsou malé a dobře viditelné. Proto je u systémů s otevřenými zdrojovými kódy paravirtualizace velice oblíbená Problém přichází u přizpůsobování operačních systémů s uzavřenými zdrojovými kódy, protože není zcela snadné je modifikovat, případně lze, ale jen velmi špatně a omezeně.[3]

2.4.2 Paravirtualizace MMU

Paravirtualizace používá při překladu adres tabulku stránek virtualizovaného systému. Virtualizovaný systém má povoleno číst záznamy v tabulce. Požaduje-li virtualizovaný systém změnu v tabulce, musí si ji u hypervizoru vyžádat. Hypervizor poté zkontroluje požadavek na změnu a zanese ji do tabulky. Tento způsob správy paměti přináší časté volání hypervizoru, to vede k častému přepínání kontextu. Proto se zde zavádí optimalizace. Virtualizovaný systém řadí požadavky do fronty a hypervizor provádí změny dávkově. Optimalizována je i práce s TLB. Paravirtualizace používá techniku, při které v každém adresovém prostoru virtualizovaného systému rezervujeme část pro kód hypervizoru. Přepínání kontextu mezi procesem, ve virtualizovaném systému, a hypervizorem pak nebude vyžadovat přepínání adresového prostoru.[3] [5]

2.4.3 Paravirtualizace I/O zařízení

Paravirtualizovaný systém využívá, oproti úplné virtualizaci, existence hypervizoru. Paravirtualizovaný stroj odevzdává I/O požadavky v abstraktní formě. Konverzi požadavků do formy přijatelné pro fyzická zařízení má na starosti hypervizor. Celková složitost zpracování požadavku se velmi sníží. Hlavní výhodou paravirtualizovaných ovladačů je, že vědomě přistupují k virtuálním zařízením.[3]

Komponenty paravirtualizovaného ovladače:

- **Syntetický ovladač**

  Má na starost prostřednictvím sdílené paměti přenos dat mezi virtualizovaným systémem a hypervizorem. Z hlediska minimalizace režie přenosu se snažíme syntetický ovladač co nejvíce zjednodušit. Operace, se kterými obyčejně vystačíme, jsou čtení, zápis nad sdílenou pamětí a rutiny pro doručení asynchronních událostí. Syntetický ovladač lze rozdělit do dvou částí. Jedna část je implementována virtualizovaným systémem (frontend) a druhá část implementuje hypervizor (backend). Hypervizor multiplexuje data jdoucí od virtualizovaných systémů na zařízení a demultiplexuje data jdoucí opačným směrem.[3]

- **Ovladač fyzického zařízení**

  Vyvíjet ovladače v rámci hypervizoru pro přímou komunikaci s hardwarem není nutné. Postačují nativní ovladače z běžných operačních systémů, které hypervizor do sebe integruje.[3]

3 Praktická část

3.1 Problematika

Jak se postupem času počítačová technika vyvíjela, přestávalo být dostačující porovnávání výkonu počítačů z hlediska technických parametrů hardwaru. Proto byly vyvinuty testy, které umožňují porovnání různých architektur. Ve výpočetní technice je program testující výkon hardwaru nazýván: benchmark. Benchmark je tedy software, který se spouští za účelem porovnání výkonnosti testovaného stroje. Pomocí benchmarkingu lze určit nejen výkonnost počítačového hardwaru, ale také softwaru, programovacích jazyků, databázových systémů. Dále lze použít k diagnostickým účelům, které jsou důležité pro odhalování závad či slabých míst systému. Benchmark je navržen tak, aby napodobil určitý druh zátěže, jak na celém systému, tak na jednotlivém hardwaru. Známe dva druhy testů, syntetické a aplikační. Syntetický benchmark je speciálně vytvořený test, který nevychází z žádných konkrétních aplikací. Výsledky syntetického testu nemusí vždy odpovídat reálnému světu. Avšak jsou vhodné pro testování jednotlivých komponent jako například pevného disku a síťového zařízení. Naopak aplikační testy testují chování systému v konkrétních aplikacích. Prodejci hardwaru často mají tendence ladit svoje produkty tak, aby obstály na nejpoužívanějších benchmarcích. Proto musíme být opatrní ohledně prezentování výsledků a při výběru testů. Mnoho benchmarků je zaměřováno na rychlost výpočetního výkonu a zanedbává další důležité funkce hardwaru. Jedná se o funkce kvality služeb. Mezi málo měřené nebo vůbec neměřené funkce například patří: zabezpečení, dostupnost, spolehlivost, integrita, provozuschopnost, škálovatelnost.

Nyní si přiblížíme problematiku testování různých virtualizačních technologií. Aby testování a následné srovnání mohlo být korektní, bylo potřeba vybrat testovací nástroj, který bude snadno dostupný. To znamená nejlépe open source, kde zdrojové kódy testů budou otevřené. Jednak se tím zajistí čitelnost testů, jejich spuštění si bude moct provést každý uživatel sám doma na svém stroji. Open-source benchmarky jsou velice oblíbené, tudíž lze po nahlédnutí do výsledků ostatních uživatelů předvídat naše výsledky měření. Dalším důležitým aspektem je zajištění, aby testy byly multiplatformní, to znamená, aby mohly běžet na různých počítačových platformách. Proto byla vybrána testovací sada Phoronix-test-suite. Phoronix může být instalován na většině dnes známých a běžně používaných operačních systémech jako je Linux,

Windows 7, OpenSolaris, Mac OS X a BSD. V našem případě byly vybrány jako hostující systémy produkty z dílen Microsoft a CentOS. Konkrétně Windows 7 Professional a CentOS 5.8. Na Windows 7 byly postupně nainstalovány virtualizační nástroje VirtualBox a VMware. Oba tyto produkty jsou zástupci plné virtualizace. CentOS 5 byl také obohacen o virtualizační software VirtualBox, VMware a představitele paravirtualizace Xen. Systém pro hosty virtualizačních nástrojů byl vybrán opět CentOS 5. Důvod výběru operačního systému pro hosty, je ten, že autor již na tomto systému pracoval a zná prostředí GNOME. Po vytvoření všech hostů na virtualizačních nástrojích, nainstalování operačního systému, bylo zapotřebí opatřit systémy testovací sadou Phoronix-test-suite a testovacími balíčky. Poté lze spouštět jednotlivé testy.[13]

3.2 Virtualizační nástroje

3.2.1 XEN

Jeden z nejznámější open source virtualizačních nástrojů od vývojářů XenSource, Inc. První verze Xenu byla vyvinuta v laboratořích na univerzitě v Cambridge. V roce 2010 se Xen stal komunitním projektem publikovaným pod licencí GPL. Xen nabízí rozhraní pro virtualizaci hardwaru a více operačních systému na jednom fyzickém stroji. Obsahuje hypervizor, který je nahrán do kruhu 0. Zde obhospodařuje operační paměť a I/O operace, které pak zprostředkovává hostům. Hypervizor po spuštění nahraje do kruhu 1 operační systém s nástrojem Xen. Ten se nazývá dom-0. Dom-0 neboli hostitel a můžeme z něj spouštět další domény, nastavovat je, migrovat atd. Operační systémy, které jsou z hostitele následně spuštěny, se označují dom-U a jejich jádra běží taktéž v kruhu 1, jako jádro hostitele. Paravirtualizace předpokládá upravené jádro jak hosta, tak hostitele. U linuxových distribucí, to není žádný problém, protože zdrojové kódy jsou zveřejněné a volně k dispozici. Pro svou bakalářskou práci jsem si vybral jako hostující systém CentOS 5 založený na linuxových distribucích Red Hat Enterprise Linux, jelikož poskytuje podporu pro Xen. Přesněji CentOS 2.6.18-308.1.1.el5xen.[9] [10]

3.2.2 VMware

Od firmy VMware, Inc. byl vybrán proprietární multiplatformní virtualizační nástroj VMware Workstation 7.1.5. pro počítače založené na architektuře x86. Jedná se o zástupce plné virtualizace umožňující spouštět na jednom fyzickém stroji více operačních systémů. VMware Workstation nabízí kompletní sadu virtualizovaného hardwaru, které v sobě obnáší procesor, operační paměť, grafické adaptéry, disky, síťové karty a mnoho dalšího. Na rozdíl od klasických emulátorů VMware neemuluje instrukční sady pro hardware, který není fyzicky shodný. V případě, že chceme přenést hosta mezi počítači používající různé instrukční sady, nastává problém. Může nastat například mezi procesory Intel a AMD. Hlavní výhodou virtuálních strojů emulovaných na VMware Workstation je přenositelnost mezi různými hostitelskými stroji. V nejvyšších řadách produktů je dokonce nabízená možnost nazývaná VMotion, která umožňuje přenášet jednoho pracujícího hosta mezi fyzickými stroji se shodnou konfigurací.[11]

3.2.3 VirtualBox

Jako dalším multiplatformním virtualizačním nástrojem zastupujícím plnou virtualizaci byl vybrán software od firmy Oracle. Stejně jako VMware Workstation je vyvíjen pro počítače architektury x86. VirtualBox lze spouštět na řadě operačních systémů a celá řada systémů na něm lze také virtualizovat. Podporované hostující systémy jsou například: Linux, Mac OS X, Windows XP,Windows Vista, Windows 7, Solaris, a OpenSolaris. Jako hosta můžeme spustit operační systémy založené na platformách: Windows, Linux, BSD, OS/2, Solaris Jádro hostujícího systému, které normálně běží v kruhu 0, je spuštěno v kruhu 1. Protože obsahuje mnoho privilegovaných instrukcí, které nemohou být prováděny nativně v kruhu 0. VirtualBox potřebuje ke správnému běhu nástroje Code Scaning a Analaysis Manager ke sledování kruhu 0. Před prvním spuštěním identifikují tyto nástroje problematické instrukce a potom je zavolán Patch Manager, který sestaví ekvivalentní bezpečné instrukce a vloží je do hypervizoru. Virtualizovaný počítač pak běží v uživatelském režimu kruhu 3.[15]

3.3 Testovací nástroje

3.3.1 Vlastní testovací nástroj

Pro zjištění časové náročnosti byla vybrána úloha k zjišťování sousednosti v síti elementů. Touto úlohou se autor zabýval částečně již v Bakalářském projektu. Elementy jsou obsaženy v síti, která má pevně dané rozměry. Počet elementů v síti je konstantní. Síť obsahuje tři druhy elementů: úsečky, trojúhelníky a čtyřstěny. Elementy jsou v síti rozmístěny náhodně a jejich pozice je definována souřadnicemi jejich bodů. Existence sousednosti byla hledána pomocí společných vrcholů různých elementů. Programovací jazyk JAVA umožnil element a jeho vlastnosti popsat jako objekt. Síť elementů potom můžeme chápat jako kolekci objektů se svými vlastnostmi a parametry. To umožňuje procházením kolekce objektů a porovnáváním parametrů, konkrétně bodů, zjišťovat sousednost.[8]

Element

Síť obsahuje tři typy elementů: úsečky, trojúhelníky a čtyřstěny. V souboru popisující síť jsou za značkou „$Elements“ definovány elementy. Na prvním místě se nachází index elementu, dále typ elementu. Na třetím místě je počet tagů (téměř vždy 3) a následují zmíněné tři tagy. První tag je fyzická entita, která náleží elementu, druhý je geometrická entita a třetí by měla vyjadřovat příslušnost k oddílu. Poslední údaje jsou body, ze kterých se element skládá.[8]

Síť elementů

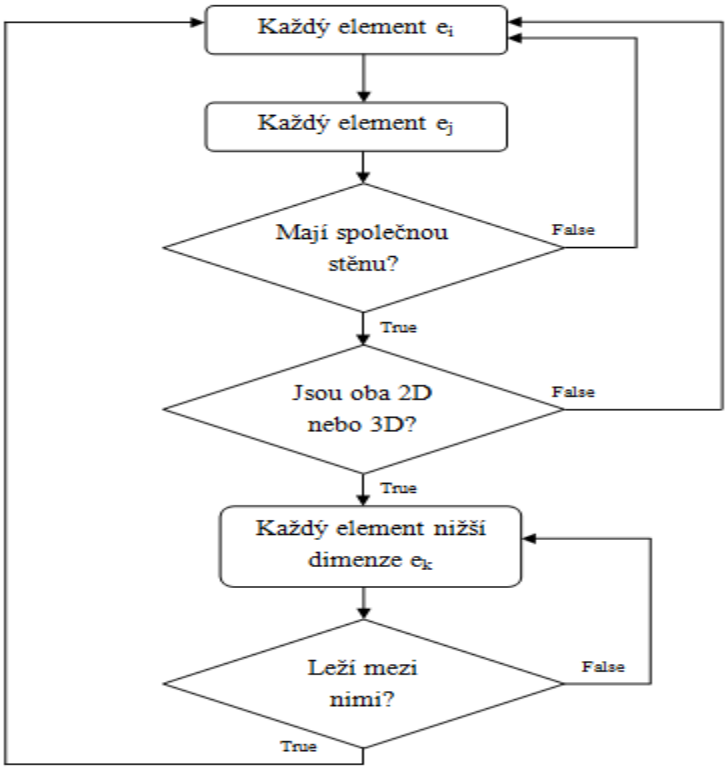
Síť Elementů je určena uzly (body, kde se elementy scházejí) a jednotlivými elementy, které jsou určeny svým typem. V souboru, který obsahuje zmíněnou síť, je za značkou „$Nodes“ počet uzlů. Uzel je definován tak že, má na prvním místě index a poté jeho souřadnice v 3D prostoru (X, Y, Z). Body jsou spojené úsečkami, ty vytváří elementy. Počet elementů ovlivňuje počet bodů v síti. Čím více bodů tím je síť hustší. Veškeré testy byly prováděny na síti, která obsahovala 2953 bodů, 14818 elementů. Z nichž bylo vytvořeno 783 trojúhelníků a 14 035 čtyřúhelníků.[8]

Sousednost

V algoritmu, který byl vybrán pro porovnání rychlosti jednotlivých virtualizačních nástrojů, hledáme v 3D poli objektů sousednost. Dvojice elementů mající stejnou dimenzi spolu sousedí v případě, že mají společnou stěnu. Dvojice elementů různé dimenze spolu sousedí pokud, element nižší dimenze tvoří stěnu elementu vyšší dimenze. Sousednost není definována mezi dvojicemi 1D a 3D. Když se objekty protínají, nejsou sousedé.[8]

Algoritmus

V prvním kroku jsou načítány ze souboru všechny body (jejich souřadnice) a pak elementy v síti, které tvoří očíslované body. Poté algoritmus vezme první element, porovná ho se všemi elementy v síti, kromě sebe sama. V případě že narazí na dva objekty typu 2D nebo 3D, pak prochází všechny elementy znovu a hledá element, který by mohl narušit sousednost. Sousednost testuje tak, že zjistí počet společných bodů. U dvojic 2D (3D) jsou potom nalezené body srovnávány s body prvků s dimenzí o 1 menší. Tento algoritmus byl vybrán kvůli jeho časové náročnosti, jednoduchosti a možnosti o rozšíření vlákny.[8]



Obr. 4 Algoritmus zjišťování sousednosti [8]

3.3.2 Phoronix-test-suite

Jako testovací nástroj, podle kterého budou porovnány virtualizační technologie, byla vybrána testovací sada s názvem Phoronix-test-suite, která je součástí Phoronix Media. Jedná se o open source testovací framework založený v roce 2004 Michaelem Labarelem, který je současně vedoucím vývojářem. Licence, kterou používá je GNU GPLv3. Phoronix-test-suite je jedna z nejrozsáhlejších testovacích a zároveň srovnávacích platforem, která je nyní k dispozici. Poskytuje možnost snadno rozšířit stávající seznam balíčků testů o nové testy, zkoušky nebo benchmarky. Phoronix-test-suite je navržen tak, aby plnil jak kvalitativní, tak i kvantitavní měřítka testů. A zároveň, aby způsob testování byl čistý a snadno použitelný. Pro sdílení a následné porovnávání testů s jinými uživateli existuje komponenta OpenBenchmarking.org, která je součástí Phoronix Media. Tato infrastruktura poskytuje veřejné, ale i soukromé sdílení výsledků. Lze zde nalézt přehled všech dostupných testů a testovacích balíčků. Dále pak výsledky ostatních uživatelů a jejich hardware, na kterém byl test spuštěn. Nachází se zde i tabulka hardwaru, který si vedl nejlépe v provedených testech.[12]

Postup testování

Na testovaném fyzickém stroji při testování běžely jen aplikace nezbytné pro běh operačního systému popř. virtualizačního softwaru. Jakékoliv další spuštěné aplikace by mohly výrazným způsoben ovlivnit výsledky testů. Jednotlivé testy byly rozděleny do sérií. Pro každý test byl zvolen počet opakování tak, aby výsledky poskytly relevantní data. Po každé testovací sérii, byl fyzický stroj restartován. Restartování fyzického stroje bylo z důvodu, aby měření bylo co nejpřesnější a nebyla do testování vnášena chyba. Jestliže spustíme test na softwaru Phoronix-test-suite, neprovede se pouze jednou. Test se provede opakovaně. Výsledná hodnota, která byla zapsána do tabulky, je průměr vícekrát spuštěného testu. Bylo zjištěno, že méně náročné testy se spouští vícekrát. Pokud bych uvedl příklad, méně časově náročný test Stream byl opakován 20×. Z tohoto důvodu jsou v tabulce uvedena tři měření daného testu u každého stroje. Test pro grafický adaptér J2D Microbenchmark se při jednom spuštění provedl 4×. V průměru byly testy při jednom spuštění provedeny 5-7krát. Celkově bylo spuštěno přes 140 testů na pěti virtuálních strojích a jednom fyzickém stroji. To znamená, že bylo spuštěno přibližně 5000 testů.

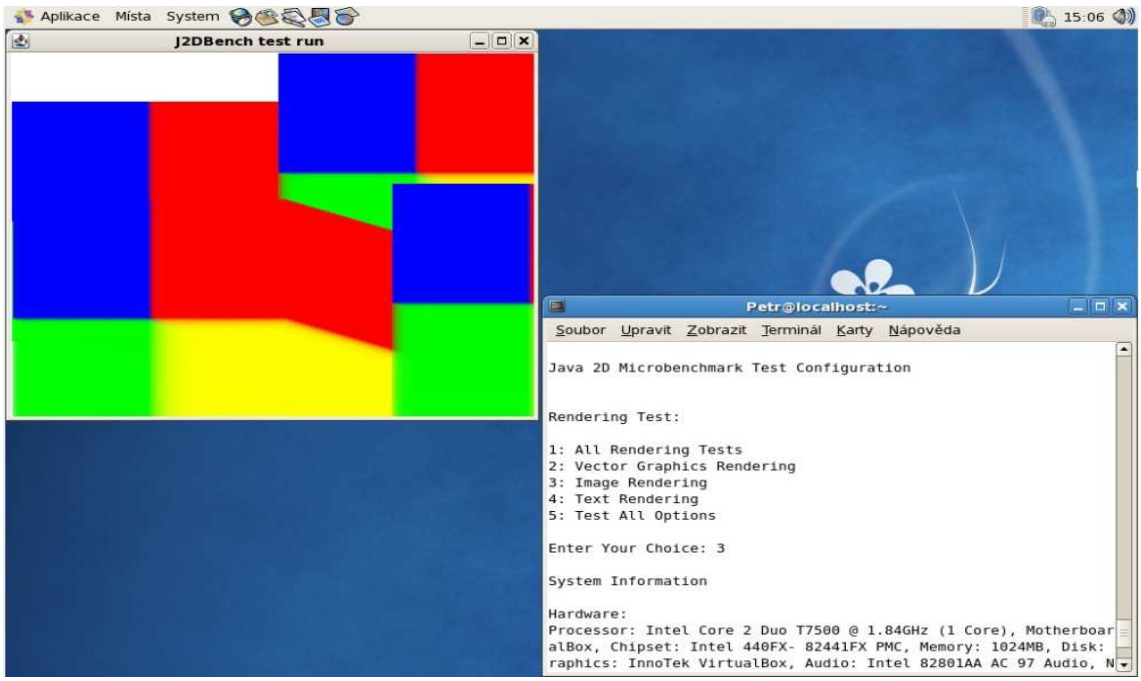
Testovaný fyzický stroj

Procesor:	Intel Core 2 Duo T5750 s frekvencí 2 GHz
Paměť:	1 x 1 GB DDR2 667 MHz, 1 x 2 GB DDR2 667 MHz
Grafická karta:	ATI Mobility Radeon HD 3650 s 512 MB vlastní paměti
Operační systémy:	Windows 7 Professional 32b, CentOS 5.8 32b

Testované virtuální stroje

Všem hostům bylo přiřazeno stejné hardwarové nastavení.

Procesor:	Intel Core 2 Duo T7500 2 GHz (1 Core)
Paměť:	1024MB
Grafická karta:	Každý nástroj emuloval svoji.
Operační systém:	CentOS 5.8 32b



Obr. 5. Ukázka pracovního prostředí Phoronix-test-suite [12]

Ukázka práce v terminálu s Phoronix-test-suite:

phoronix-test-suite – vypíše přehled možností, základní informace o produktu

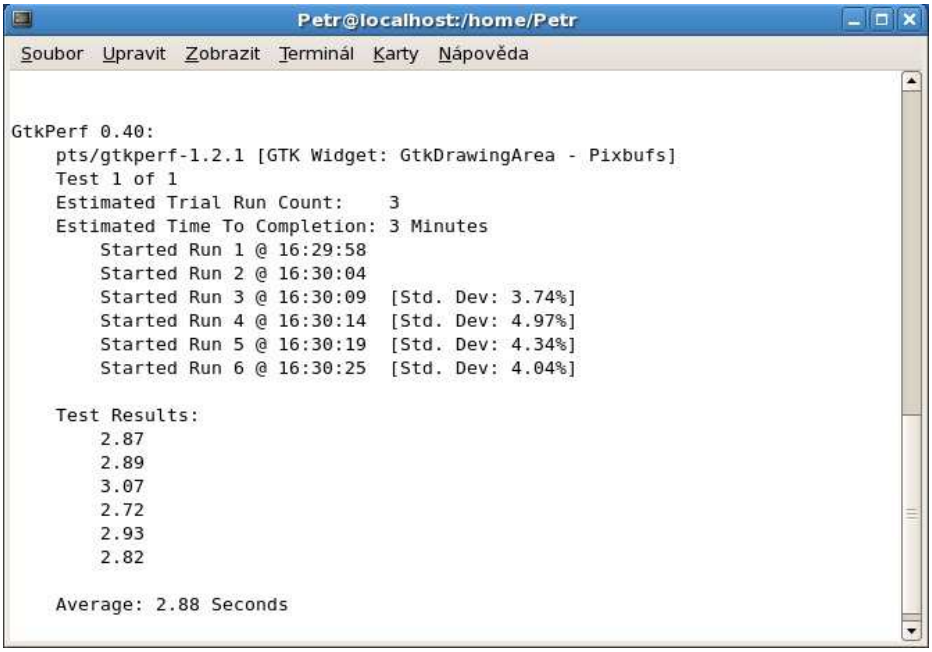
phoronix-test-suite list-available-suites/tests – vypíše seznam dostupných balíčků/testů

phoronix-test-suite install Object – nainstaluje vybraný test

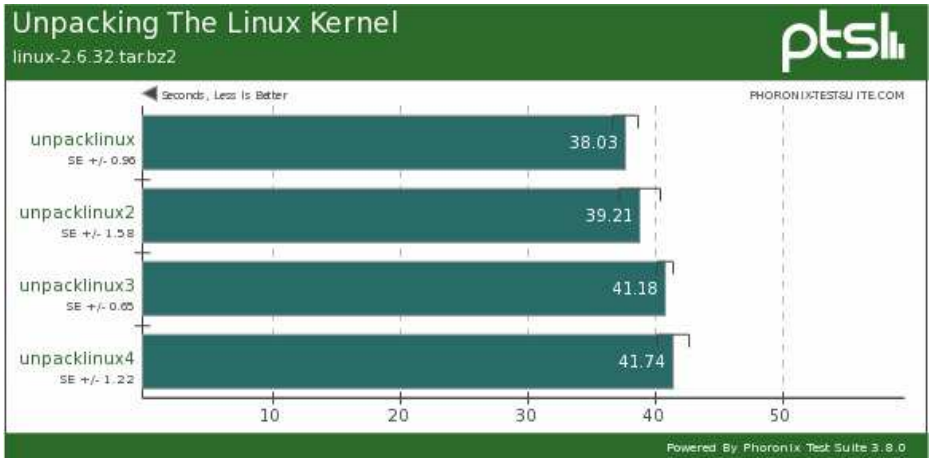
phoronix-test-suite benchmark Object – spustí test, s možností uložení výsledku

phoronix-test-suite list-saved-results – vypíše uložené výsledky testů

phoronix-test-suite show-result Object – vypíše výsledky testů vybraného objektu



Obr. 6. Ukázka výpisu výsledku testu GtkPerf [12]



Obr. 7. Ukázka grafu generovaného nástrojem Phoronix-test-suite [12]

Použité testovací komponenty sady Phoronix

Aby bylo testování komplexní a multiplatformní, byly vybrány sady testů, které byly postupně spouštěny na jednotlivých virtuálních strojích. To znamená, procesor, operační paměť, grafický adaptér a pevný disk.

Pro procesor byly vybrány testy:

pts/compression - Tato testovací sada se skládá z testů, která měří různé formy komprese souborů jako je gzip, bzip2, 7zip a lzma. Sada používá stejný soubor o velikosti 512MB. Velice vhodná pro testování procesoru, operační paměti a celkového výkonu systému. Výstupem je čas potřebný pro kompresi.

pts/himeno - Tento test je lineárním řešitelem tlaku Poisson pomocí Jacobi metody. Jedná se o složitý výpočet. Výstupem je hodnota v jednotkách MIPS.

Pro pevný disk byly vybrány testy:

pts/unpack-linux - Tento test měří, jak dlouho trvá rozbalení souboru linux-.tar.bz2. Velikost tohoto souboru byla v našem případě 70MB. Výstupem je čas potřebný pro rozbalení souboru.

pts/apache - Tento testovací profil měří, kolik požadavků za sekundu je schopen http server vyřídit při zátěži. Požadavků je700 000 z toho 100 se provádí souběžně. Výstupem je číslo udávající vyřízený požadavek za sekundu.

pts/aio-stress - AIO stress je asynchronní I/O benchmark vytvoření SuSe. Používá 2048MB testovací soubor**.** Výstupem je hodnota v jednotkách MB/s.

Pro operační paměť byly vybrány testy:

pts/ramspeed - Tento test testuje výkon operační paměti. Výstupem je hodnota v MB/s.

pts/stream - Stream je jednoduchý syntetický benchmark, který měří propustnost operační paměti a odpovídající výpočetní rychlost jednoduchých vektorových jader. Výstupem je hodnota v MB/s.

Pro grafický adaptér byly vybrány testy:

glxgears - Jednoduchý nástroj, který je snad v každé distribuci linuxu. Zobrazuje sadu rotujících ozubených kol a vypisuje do terminálu v pravidelných 5s intervalech počet snímků za sekundu. Velice populární srovnávací nástroj.

pts/j2dbench - Java 2D Microbenchmark. Tato sada obsahuje sérii mikro testů zjišťujících výkon OpenGL. Tento průmyslový standard specifikuje multiplatformní rozhraní (API) pro tvorbu aplikací počítačové grafiky. Výstupem je hodnota v jednotkách za sekundu.

pts/gtkperf - Tato sada obsahuje testy zaměřené na testování grafického adaptéru. Gtkperf testuje práci uživatele s grafickým uživatelským rozhraním operačních systémů. Vypočítává průměrnou snímkovou frekvenci GTK operací.[12]

4 Výsledky

4.1 Tabulky

V tabulce jsou uvedeny zpracované hodnoty všech testů ze sady Phoronix. Testů bylo provedeno celkem 18 na každém stroji. V prvním sloupci z leva jsou uvedeny názvy testů. Ve druhém a třetím sloupci jsou znázorněny výsledné hodnoty virtualizačních nástrojů VirtualBox a VMware nainstalované na operačním systému Windows 7. Ve čtvrtém až šestém sloupci jsou uvedeny virtualizační nástroje, které byly nainstalovány na operační systém CentOS. Poslední sedmý sloupec obsahuje naměřené hodnoty fyzického stroje bez použití virtualizačních nástrojů. Kompletní výsledky jsou k nahlédnutí v příloze B.

Tab. 1. Výsledné hodnoty testů sady Phoronix

	Windows		CentOS			
Nástroj	VirtualBox	VMware	VirtualBox	VMware	Xen	Normal
pbzip2	76,84	75,58	80,1	69	74,35	36,75
lzma	338,92	322,34	315,98	314,81	312,93	289,15
7zip	1449,5	1689	1374,5	1729,5	1791,5	2890
gzip	62,53	45,68	60,81	38,72	48,26	21,48
himeno	468,78	491,53	486,49	519,67	503,22	530,98
ramspeed	2474,85	2816,18	2567,66	2722,1	2926,56	2547,83
stream-copy	3111,29	2875,57	5389,85	3055,05	3204,3	3058,51
stream-scale	3013,46	2806,26	5376,53	2982,1	3147,45	2992,53
stream-add	3362,9	3185,21	6599,71	3586,71	3806,31	3516,06
aiostress	22,1	39,64	10,32	26,69	6,97	38,61
unpack-linux	39,86	42,08	35,12	35,65	26,83	21,15
apache	545,94	2735,31	548,48	2852,43	4241,42	5793,68
j2dbench	131510,8	117143,9	136825,4	122698,4	143759,6	166395,3
glxgears	636	993	799	799	1366	1461
gtkcombobox	95,04	118,22	115,81	128,09	95,15	51,55
gtktbutton	33,34	45,14	37,99	50,05	33,47	17,67
gtkpixbuffs	2,5	2,86	2,73	2,77	3,41	2,1
gtkdrawning	46,89	59,83	84,4	68,46	57,99	34,82

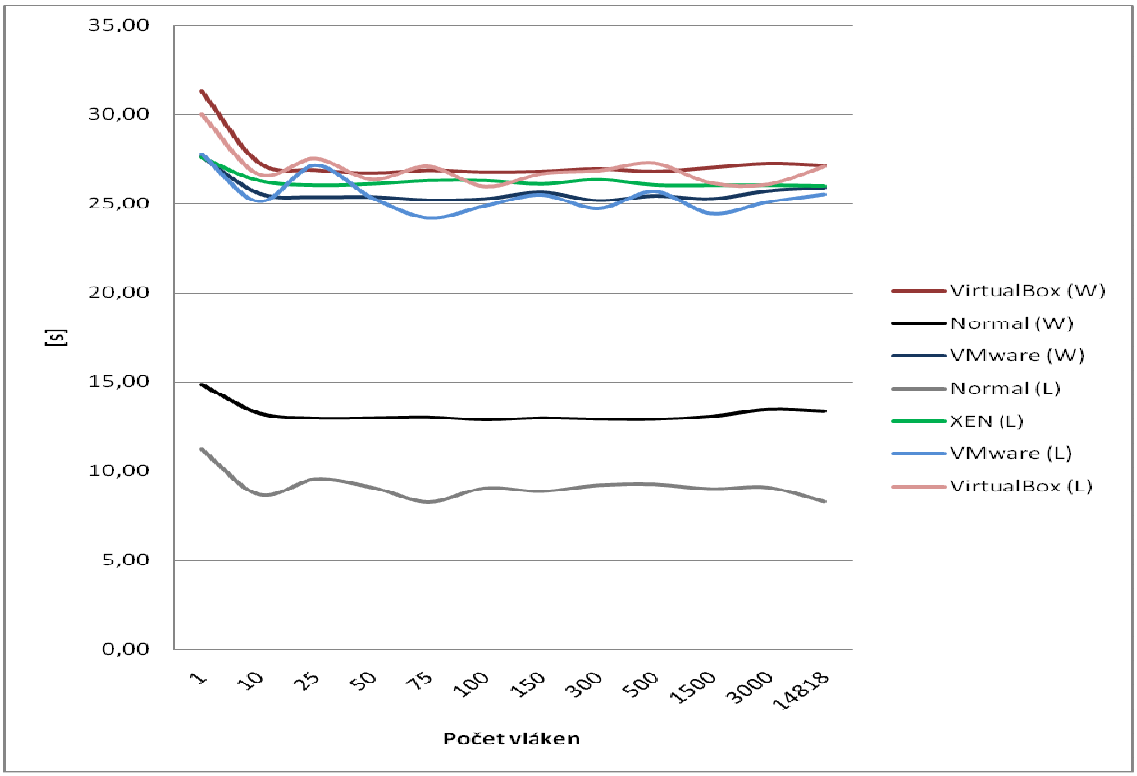
V tabulce jsou uvedeny naměřené hodnoty vlastního testovacího nástroje při použití algoritmu s vlákny.

Tab. 2. Výsledné hodnoty testů vlastního testovacího nástroje

	Windows			CentOS			
Počet vláken	VirtualBox	VMware	Normal	VirtualBox	VMware	XEN	Normal
1	31,34	27,60	14,86	29,99	27,79	27,59	11,24
10	27,29	25,58	13,25	26,67	25,14	26,29	8,70
25	26,88	25,38	12,96	27,52	27,17	26,03	9,56
50	26,71	25,39	12,97	26,38	25,34	26,10	9,11
75	26,87	25,22	13,02	27,11	24,21	26,29	8,27
100	26,76	25,28	12,86	25,97	24,87	26,29	9,06
150	26,80	25,63	12,97	26,69	25,51	26,10	8,88
300	26,95	25,19	12,90	26,86	24,73	26,35	9,23
500	26,80	25,45	12,89	27,27	25,70	26,04	9,28
1500	27,02	25,28	13,06	26,17	24,46	26,02	9,02
3000	27,24	25,71	13,46	26,09	25,10	26,03	9,10
14818	27,13	25,88	13,36	27,10	25,53	25,98	8,30
Průměr	27,32	25,63	13,21	26,98	25,46	26,26	9,15

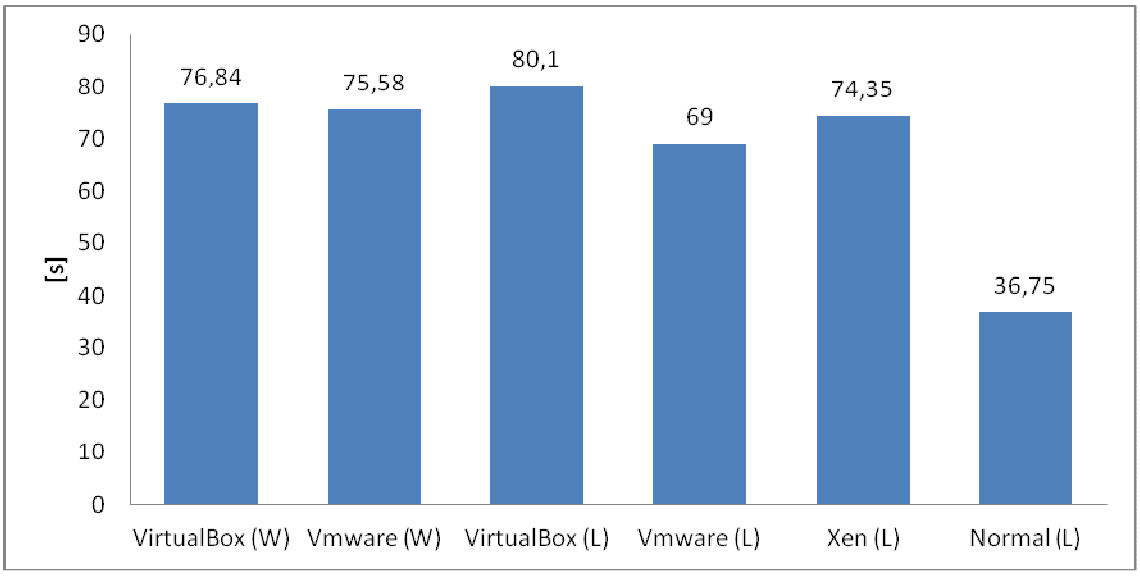
4.2 Grafy

Graficky znázorněné naměřené hodnoty vlastního testovacího nástroje při použití algoritmu s vlákny.



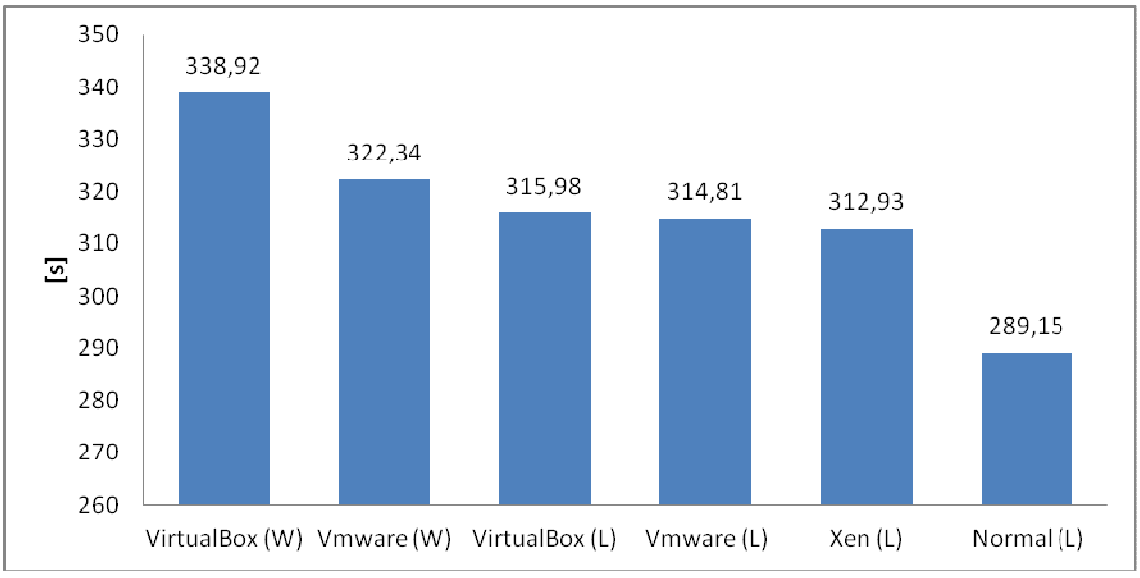
Graf 1. Časová náročnost vlastního testovacího nástroje v závislosti na počtu vláken

Graficky znázorněné naměřené hodnoty benchmarku pbzip2 testujícího výkon procesoru. V tomto testu je z virt. strojů nejvýkonnější VMware na operačním systému CentOS. XEN je o 5,35 vteřiny méně výkonný než VMware. Rozdíl mezi VMware na CentOS a fyz. strojem je téměř dvojnásobný v neprospěch virtualizace.



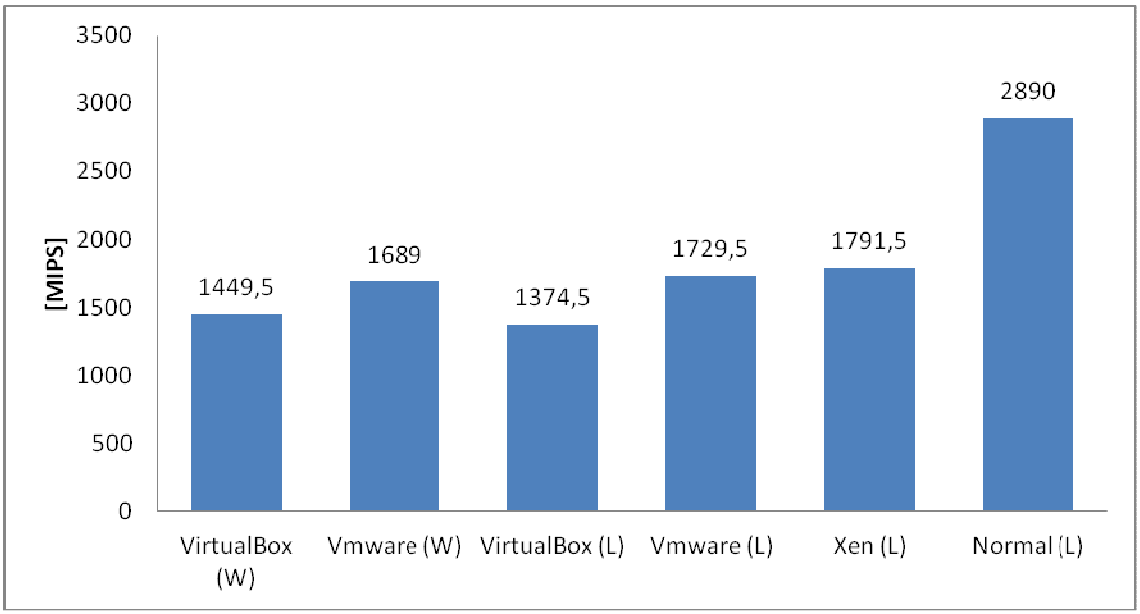
Graf 2. Časová náročnost benchmarku pbzip2

Graficky znázorněné naměřené hodnoty benchmarku komprese lzma testujícího výkon procesoru virtuálních strojů. Jako nejvýkonnější virtualizační nástroj byl vyhodnocen v tomto testu XEN. Jako druhý VMware na CentOS operačním systému. Nejvýkonnější virtualizační nástroj je o 8,2 % méně výkonný než fyzický stroj.



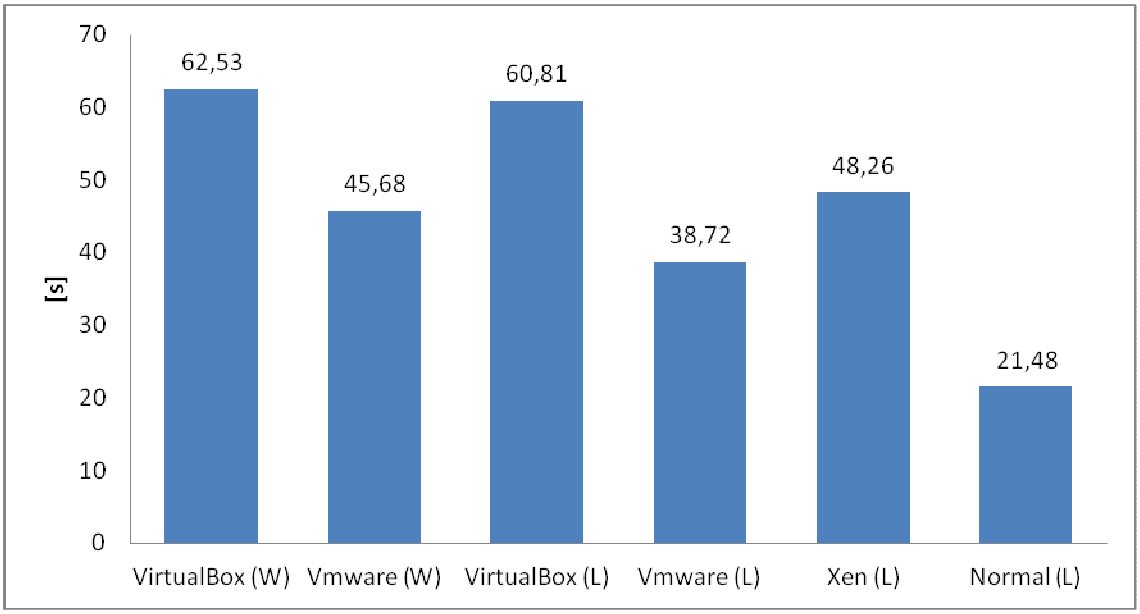
Graf 3. Časová náročnost benchmarku lzma

Graficky znázorněné naměřené hodnoty benchmarku komprese 7zip testujícího výkon procesoru virtuálních strojů. Nejvýkonnější virt. nástroj byl v tomto testu vyhodnocen XEN se 1791,5 MIPS. Jako druhý VMware na operačním systému CentOS. Nejvýkonnější virtualizační nástroj je o 38 % méně výkonný než fyzický stroj.



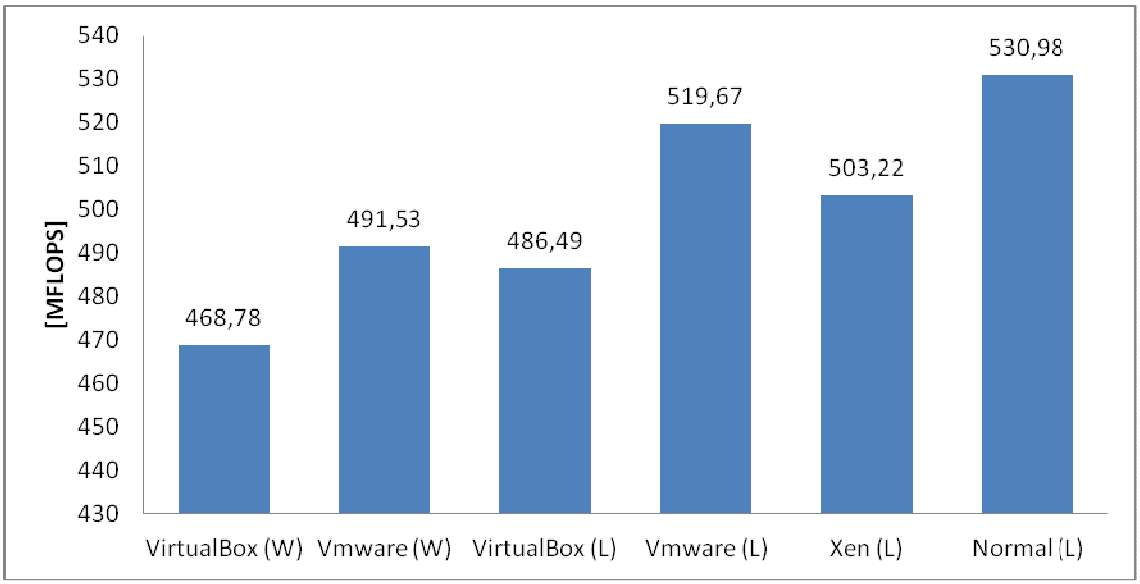
Graf 4. Náročnost benchmarku 7zip

Graficky znázorněné naměřené hodnoty benchmarku komprese gzip testujícího výkon procesoru virtuálních strojů. V tomto testu byl vyhodnocen jako nejvýkonnější virt. nástroj VMware na operačním systému CentOS, jako druhý VMware na Windows 7. Nejvýkonnější virtualizační nástroj je o 80 % méně výkonný než fyzický stroj.



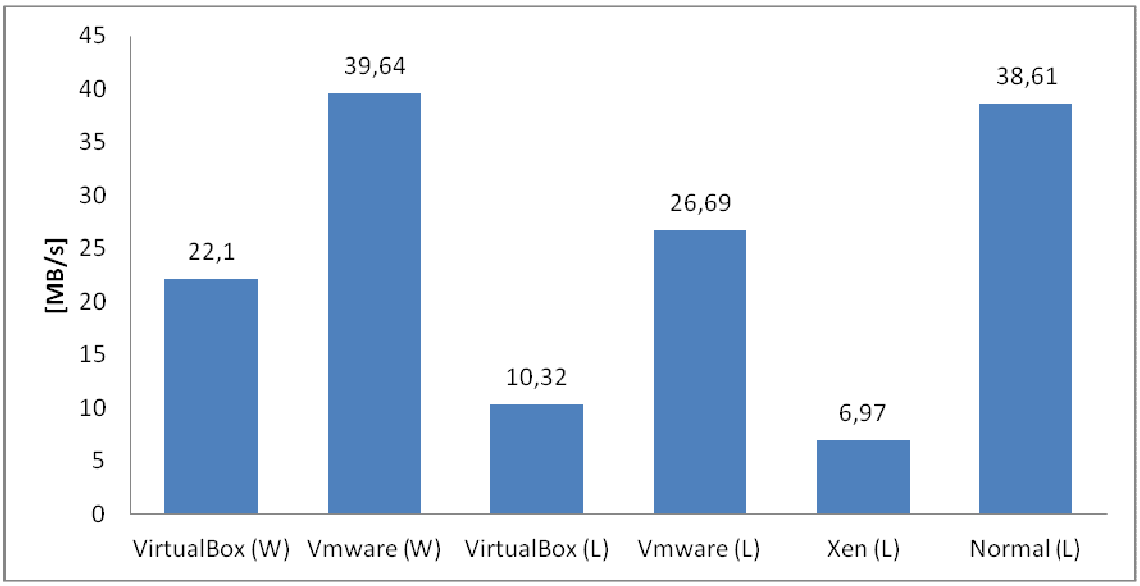
Graf 5. Náročnost benchmarku gzip

Graficky znázorněné naměřené hodnoty benchmarku komprese Himeno testujícího výkon procesoru virtuálních strojů. Nejvýkonnějším virt. nástrojem byl v tomto testu VMware na CentOS. Jako druhý XEN. Nejvýkonnější virtualizační nástroj je o 2 % méně výkonný než fyzický stroj.



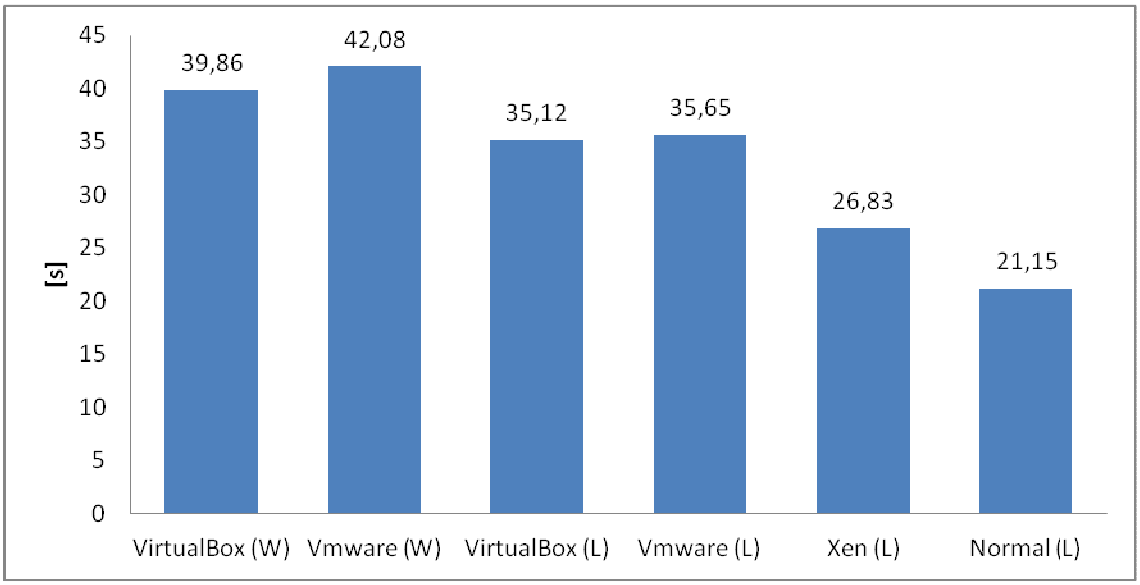
Graf 6. Náročnost benchmarku Himeno

Graficky znázorněné naměřené hodnoty benchmarku Aio-stress testujícího výkon I/O operací pevného disku virtuálních strojů. V tomto testu dosáhl nejvyššího výkonu VMware na Windows 7. Druhý byl fyzický stroj. VMware je o 2,5 % výkonnější než fyzický stroj. Jeden z testů, kde XEN skončil nejhůře.



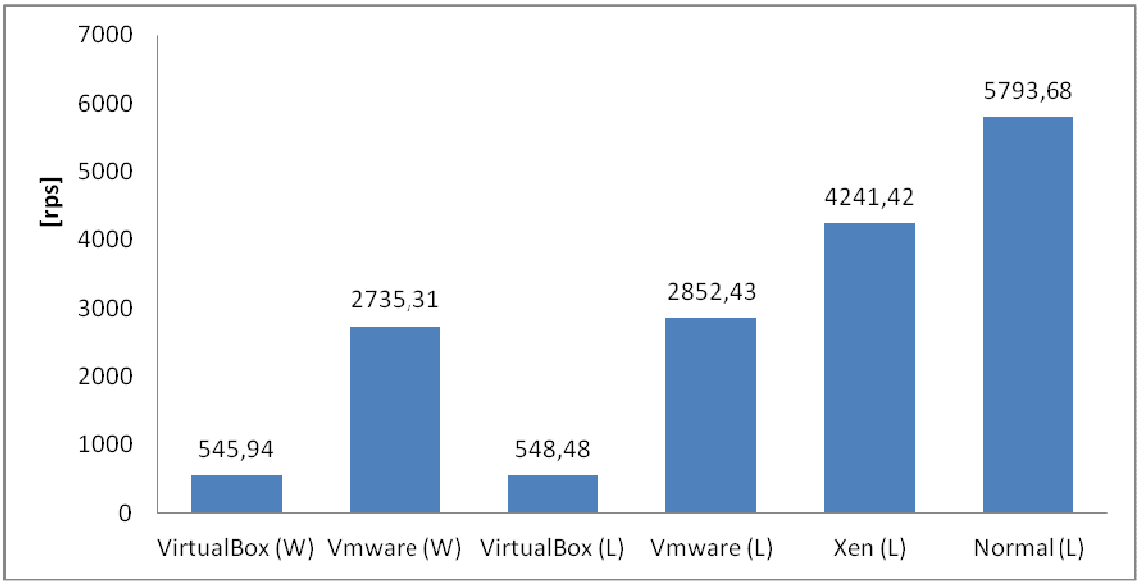
Graf 7. Náročnost benchmarku Aio-stress

Graficky znázorněné naměřené hodnoty benchmarku Unpack-linux testujícího výkon pevného disku virtuálních strojů. Paravirtualizovaný XEN byl vyhodnocen v tomto testu jako nejvýkonnější. Druhý byl VirtualBox na CentOS. Nejvýkonnější virtualizační nástroj je o 27 % méně výkonný než fyzický stroj.



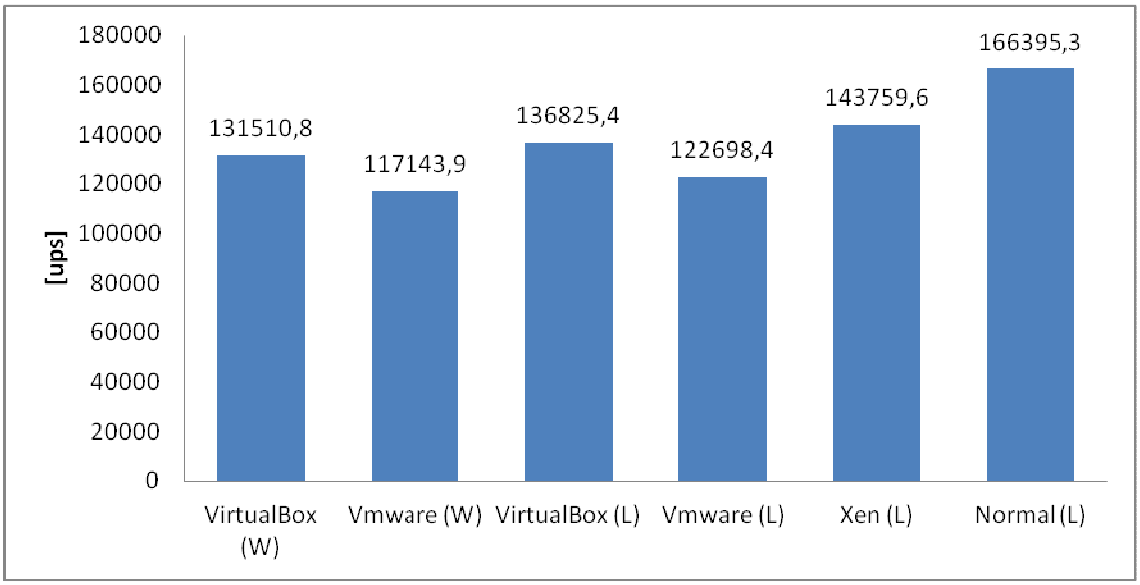
Graf 8. Časová náročnost benchmarku Unpack-linux

Graficky znázorněné naměřené hodnoty benchmarku Apache testujícího výkon pevného disku virtuálních strojů. Jako v předchozím testu byl i zde vyhodnocen nejlépe paravirtualizovaný XEN. Druhý VMware na CentOS. Rozdíl mezi nejrychlejším virtualizačním nástrojem a fyzickým strojem činil 27 % v neprospěch XENu.



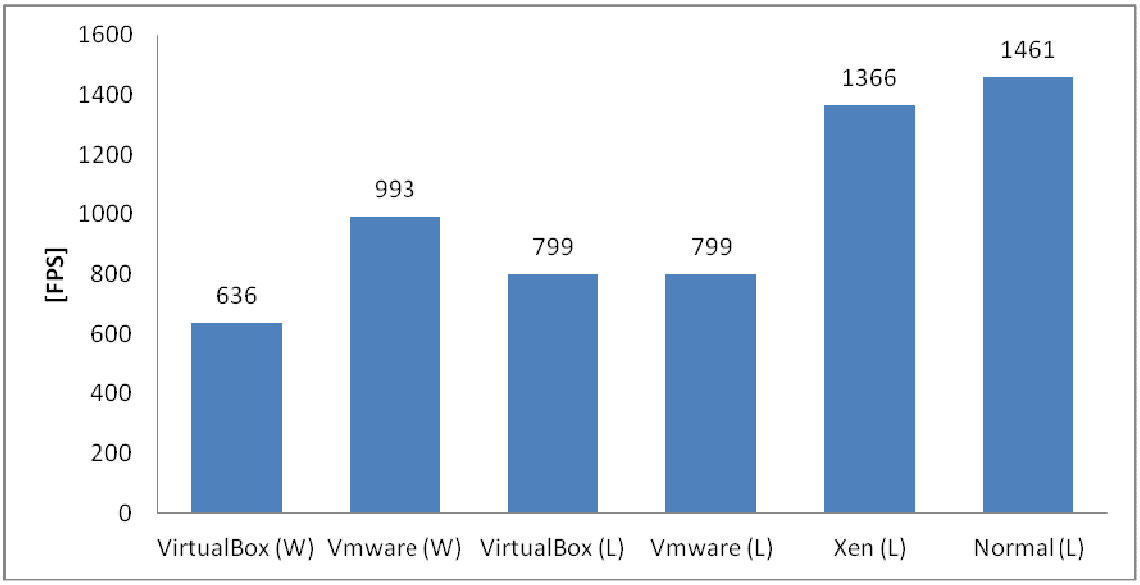
Graf 9. Časová náročnost benchmarku Apache

Graficky znázorněné naměřené hodnoty benchmarku Java 2D Microbench testujícího výkon grafického adaptéru virtuálních strojů. Jako nejvýkonnější virt. nástroj při použití J2D testu byl vyhodnocen nástroj XEN. Druhý VirtualBox na CentOS. Virtualizační nástroj XEN je v tomto testu o 13,5 % méně výkonný než fyz. stroj.



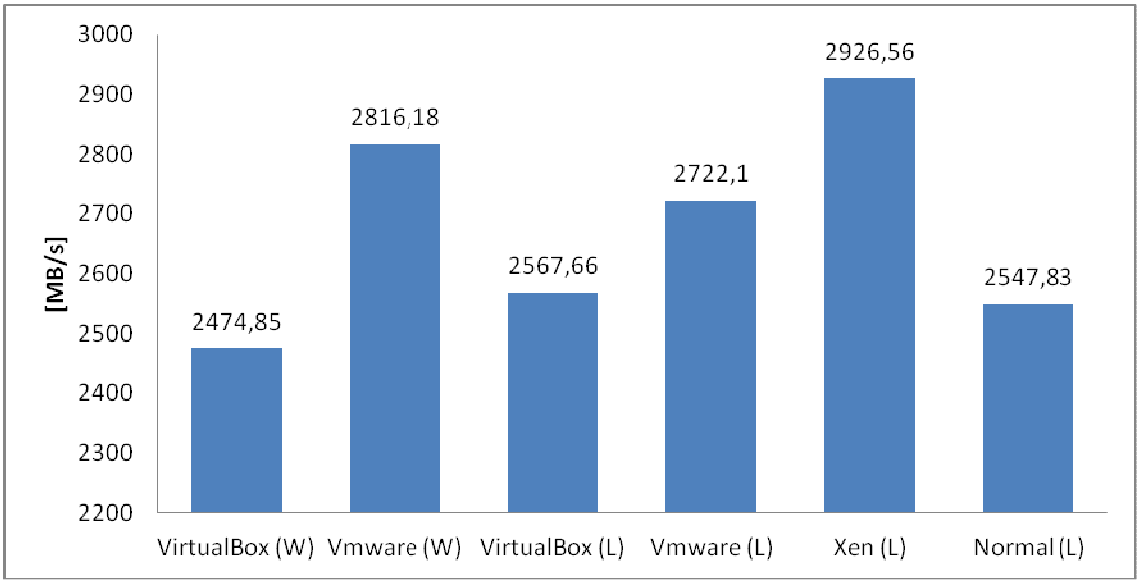
Graf 10. Náročnost benchmarku Java 2D Microbench

Graficky znázorněné naměřené hodnoty benchmarku Glxgears testujícího výkon grafického adaptéru virtuálních strojů. V tomto testu byl XEN vyhodnocen jako nejvýkonnější. Druhý VMware na Windows 7. Virtualizační nástroj XEN v tomto testu byl o 6,5 % méně výkonný než fyzický stroj.



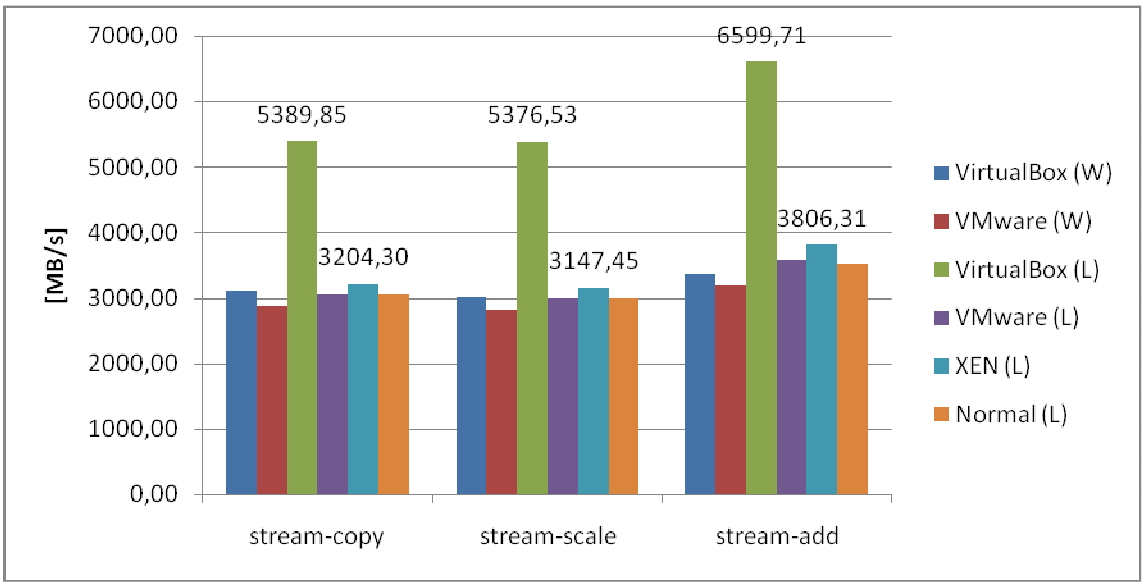
Graf 11. Náročnost benchmarku Glxgears

Graficky znázorněné naměřené hodnoty benchmarku Ramspeed testujícího výkon operační paměti virtuálních strojů. Nejvýkonnější paravirtualizovaný nástroj XEN, je o 15 % výkonnější než fyzický stroj. Druhý VMware na Windows 7 je o 10 % výkonnější než fyzický stroj.



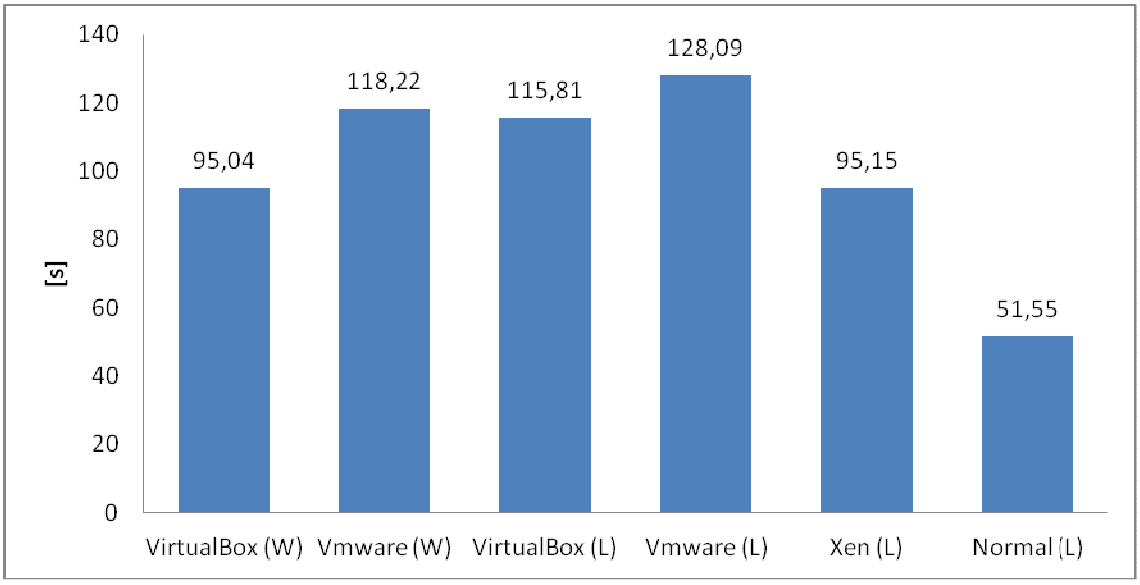
Graf 12. Náročnost benchmarku Ramspeed

Graficky znázorněné naměřené hodnoty benchmarků Stream-copy, Stream-scale a Stream-add testující výkon operační paměti virtuálních strojů. Nejvýkonnější virt. nástroj v tomto testu byl vyhodnocen VirtualBox na CentOS. Se svými výsledky je v průměru o 80 % výkonnější než fyzický stroj. Druhý skončil celkově XEN, v průměru o 5 % výkonnější n než fyz. stroj.



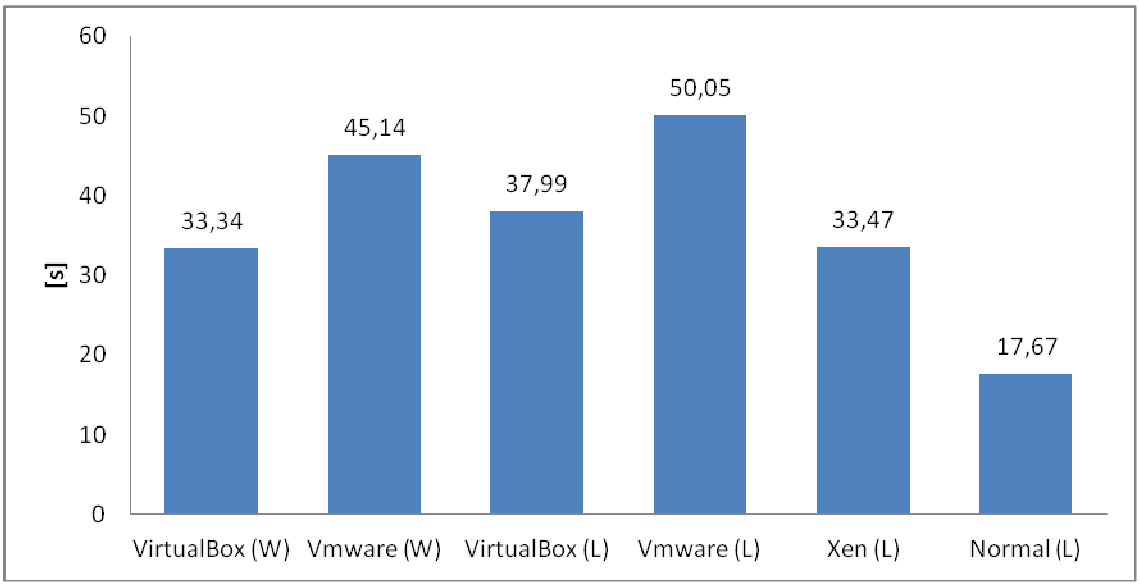
Graf 13. Náročnost sady benchmarků Stream

Graficky znázorněné naměřené hodnoty benchmarku Gtk-ComboBox testujícího výkon grafického adaptéru virtuálních strojů. Srovnatelné výsledky testů řadí na první místo dva nástroje VirtualBox na Windows 7 a XEN, které jsou o 84 % méně výkonné než fyzický stroj. Druhý VirtualBox na CentOS je o 124 % méně výkonný než fyzický stroj.



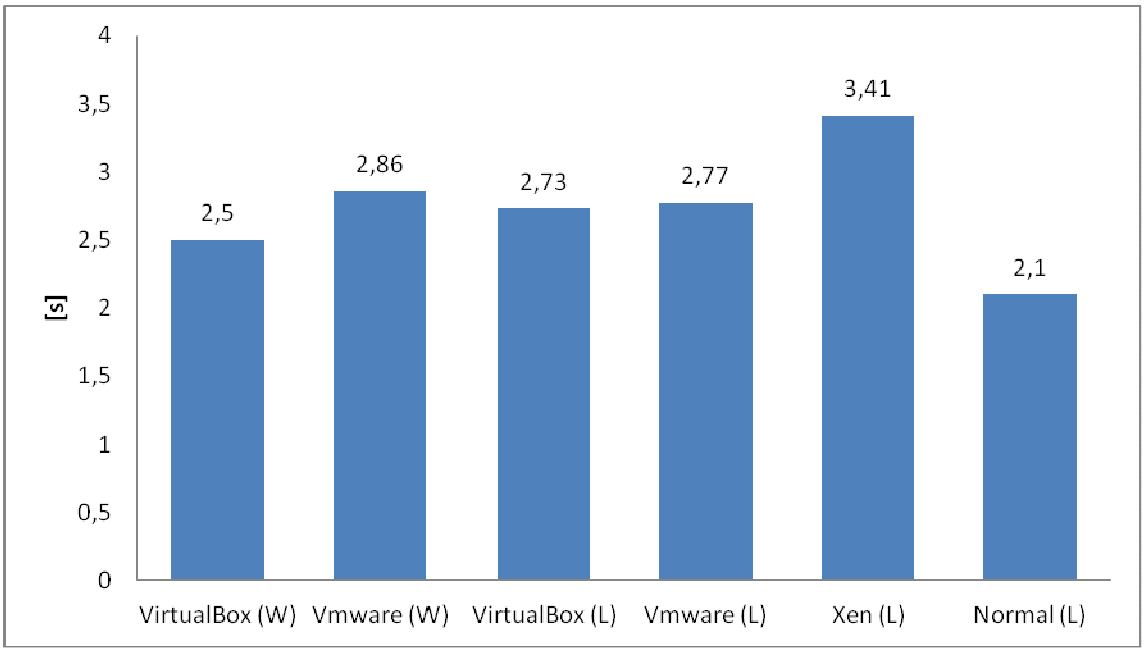
Graf 14. Náročnost benchmarku Gtk-ComboBox

Graficky znázorněné naměřené hodnoty benchmarku Gtk-Buton testujícího výkon grafického adaptéru virtuálních strojů. Obdobný trend jako u předchozího testu můžeme vidět zde. VirtualBox na Windows 7 a XEN jsou zástupci nejvýkonnějších virt. strojů v tomto testu. Avšak jsou o 88 % méně výkonné než fyzický stroj. Druhé místo VirtualBox na Windows 7.



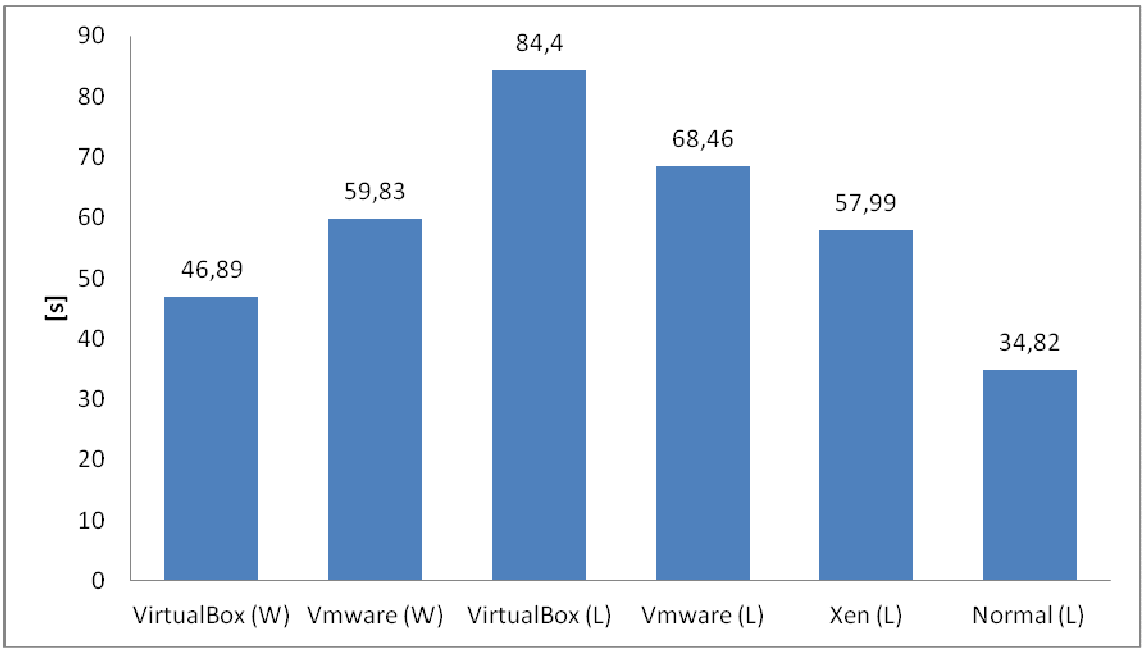
Graf 15. Náročnost benchmarku Gtk-Button

Graficky znázorněné naměřené hodnoty benchmarku Gtk-PixBuffs testujícího výkon grafického adaptéru virtuálních strojů. Nejvýkonnější virt. nástroj VirtualBox ve Windows 7. Oproti fyzickému stroji o 19 % méně výkonný. Druhý VirtualBox na CentOS. Toto je jeden ze dvou testů, kde XEN skončil nejhůře z testovaných strojů.



Graf 16. Náročnost benchmarku Gtk-PixBuffs

Graficky znázorněné naměřené hodnoty benchmarku Gtk-DrawningArea testujícího výkon grafického adaptéru virtuálních strojů. Nejlépe si s tímto testem poradil virtuální stroj VirtualBox ve Windows 7. VirtualBox je o 34 % méně výkonný než fyzický stroj. Na druhém místě paravirtualizovaný XEN.



Graf 17. Náročnost benchmarku Gtk-DrawningArea

Zhodnocení výsledků

V tabulce je uvedeno pořadí virt. strojů v jednotlivých testech.

Tab. 3. Výsledky pořadí virtuálních strojů v jednotlivých testech

		Windows	Linux	Windows	Linux	Linux
Test	č. testu	VirtualBox		VMware		XEN
Pbzip	1	4	5	3	1	2
Lzma	2	5	3	4	2	1
7zip	3	4	5	3	2	1
Gzip	4	5	4	2	1	3
Himeno	5	5	4	3	1	2
Aiostress	6	3	4	1	2	5
Unpack-linux	7	4	2	5	3	1
Apache	8	5	4	3	2	1
J2D	9	3	2	5	4	1
Glxgears	10	4	3	2	3	1
Ramspeed	11	5	4	2	3	1
Stream-copy	12	3	1	5	4	2
Stream-scale	13	3	1	5	4	2
Stream-add	14	4	1	5	3	2
Gtk-combobox	15	1	3	4	5	2
Gtk-button	16	1	3	4	5	2
Gtkpixbuff	17	1	2	4	3	5
Gtk drawning	18	1	5	3	4	2
Medián		4	3	3,5	3	2
Průměr		3,388889	3,111111	3,5	2,888889	2

V tabulce je uvedeno pořadí hardwarových komponent virt. strojů v jednotlivých testech.

Tab. 4. Výsledky pořadí hardwaru na jednotlivých virtuálních strojích

	Windows 7	CentOS	Windows 7	CentOS	CentOS
	VirtualBox		VMware		XEN
Procesor medián	5	4	3	1	2
Procesor průměr	4,6	4,2	3	1,4	1,8
Pevný disk medián	4	4	3	2	1
Pevný disk průměr	4	3,33	3	2,33	2,33
RAM medián	3,5	1	5	3,5	2
RAM průměr	3,75	1,75	4,25	3,5	1,75
Graf. adaptér medián	1	3	4	4	2
Graf. adaptér průměr	1,83	3	3,66	4	2,16
Průměr	3,46	3,03	3,61	2,71	1,88

Ve vlastním testovacím nástroji skončil jako nejvýkonnější na základě naměřených dat virtualizační nástroj VMware na CentOS a Windows. Jako druhý se umístil paravirtualizační nástroj XEN. VirtualBox si v tomto testu vedl nejhůře a skončil jako poslední, bez rozdílu na jakém operačním systému byl nainstalován.
Vývoj trendu křivek, které můžeme vidět v Grafu 1, je klesající. Tudíž můžeme tvrdit, že při zvyšujícím se počtu vláken klesá časový interval, ve kterém dojde k dokončení algoritmu. Vlákna mají tedy pozitivní vliv na časovou náročnost vlastního testovacího nástroje.

Jako nejvýkonnější virtualizační nástroj na základě umístění v jednotlivých testech sady Phoronix se umístil paravirtualizační nástroj XEN. V osmnácti spuštěných testech se ze všech virtualizačních nástrojů průměrně umístil na druhém místě. Ve dvou testech skončil XEN jako nejméně výkonný. Bylo to v testech Aiostress a Gtkpixbuffs. Z hlediska hardwarových komponent skončil Xen v testech průměrně na 1,88 místě. A to i přesto, že v testu Aiostress skončil nejhůře. Je možné, že XEN v testu Aiostress prováděl velké množství neoptimalizovaných instrukcí nebo také test mohl být příliš syntetický, tedy neodpovídající reálnému provozu informačních systémů. Xen měl jeden z nejvyšších výsledků v testech na RAM paměť, zde se umístil shodně s virt. nástrojem VirtualBox. Jsou tak nejvýkonnějšími představiteli testů zaměřujících se na paměť RAM.

Virtualizační nástroj VMware nainstalovaný na operačním systému CentOS skončil v testech na druhém místě. Průměrně obsadil ve všech testech 2,88 místo. Přihlédneme-li k určování pořadí pomocí mediánu, obsadil spolu s nástrojem VirtualBox nainstalovaným na CentOS třetí místo. Testy, kde nástroj VMware skončil na první pozici jsou Pbzip a Himeno. Naopak nejhůře si vedl v testech na testování grafického prostředí Gtk-combobox a Gtk-button.

Virtualizační nástroj VirtualBox nainstalovaný na operačním systému CentOS skončil celkově na třetím místě. Průměrně obsadil ve všech testech 3,11 místo. Medián tento nástroj spolu s VMware na CentOS řadí spolu na druhé místo. VirtualBox dominoval v testech Stream, které byly určeny pro testování operační paměti. Zde vysoko překonal svým výkonem i fyzický stroj. Naopak nejhůře si vedl v testech Pbzip, 7zip a Gtk-drawning. Z hlediska hardwarových komponent dominoval nástroj VirtualBox v testech pro paměť RAM jak již bylo zmíněno. Nejhorší výkonnostní výsledky byly zaznamenány v testech pro procesor.

Jako průměrně čtvrtý se umístil virtualizační nástroj VirtualBox spouštěný na operačním systému Windows 7. Průměrně se ve všech testech umístil na 3,38 místě. Medián tento nástroj hodnotí 4 místem. VirtualBox nainstalovaný na Windows 7 si nejlépe poradil s testy Gtk, testující GUI. Ve všech čtyřech testech skončil jako nejvýkonnější nástroj. Průměrné umístění nástroje VirtualBox na 1,83 místě, zařazuje tento nástroj na místo nejvýkonnějšího nástroje z hlediska grafického adaptéru. Nejméně výkonné výsledky byly zaznamenány v celkem pěti testech a to: Lzma, Gzip, Himeno, Apache a Ramspeed. Z hlediska hardwarových komponent byly zde naměřeny nejmenší výsledky u testů procesoru.

Jako poslední virtualizační nástroj podle průměrného umístění v testech skončil VMware na operačním systému Windows 7. Jediný test, ve kterém skončil nejlépe byl Aiostress. Výsledky z tohoto testu avšak nelze brát jako příliš relevantní, jak zde již bylo zmíněno, test je nejspíš příliš syntetický. Jako nejméně výkonný skončil v testech Unpack-linux, J2D a v testovací sadě Stream. Tyto výsledky z hlediska hardwarových komponent řadí nástroj k nejméně výkonným v testech na grafický adaptér a paměť RAM.

Závěr

Cílem bakalářské práce bylo porovnat virtualizační technologie. Srovnat jejich technické vlastnosti, navrhnout jejich praktické srovnání a to zdůvodnit. Pro porovnání byla vybrána testovací sada Phoronix-test-suite a vytvořen vlastní testovací nástroj. Z této sady byly vybrány testy pro jednotlivé hardwarové komponenty počítače. Testy byly následně spouštěny na virtuálních strojích. Virtuální stroje byly vytvořeny ve virtualizačních nástrojích VirtualBox, VMware Workstation a XEN. Virtualizační nástroje byly nainstalovány na dvou operačních systémech Windows 7 a CentOS.

Virtualizace je oblíbená technika pro vývoj a ladění celých operačních systémů tak i aplikací. Velké uplatnění nachází také v oblasti serverové techniky. Možnost spouštět operační systém ve virtuálním stroji nabízí mnoho výhod. Systém lze bezpečně testovat, ladit a kompilovat. Virtualizace avšak nemá jen kladné vlastnosti, cenou za tyto vyslovené výhody je výkon. Virtuální stroj nenabízí stejné výkonnostní prostředky jako fyzický stroj. Na internetu se nachází celá řada virtualizačních softwarů. Jedná se o software vyvíjený jak velkými firmami, tak malou skupinou vývojářů. Vybrat mezi produkty z hlediska výkonnostních specifik není lehké. Výrobci se často snaží své produkty prosazovat svými výsledky testů, které jsou speciálně navrženy pro jejich produkty.

V této bakalářské práci bylo provedeno celkem osmnáct jedinečných testů ze sady Phoronix-test-suite. Tyto testy zahrnují komplexní testování důležitých komponent počítače. Dále byl vytvořen vlastní testovací nástroj, používající algoritmus určování sousednosti v síti elementů za použití vláken. Řešení algoritmu pomoci vláken nám umožňuje nahlédnout na změny výkonnostních průběhů, jestliže máme možnost volby počtu běžících vláken.

Možností, jak srovnávat virtualizační technologie, je celá řada. V této bakalářské práci byl vybrán způsob testování výkonnostních specifik. V celkovém hodnocení sady Phoronix vyšel jako nejvýkonnější paravirtualizační nástroj XEN. Na druhém místě nástroj VMware a jako poslední nástroj VirtualBox. Z výsledků vlastního testovacího nástroje vychází nástroj VMware jako nejvýkonnější. Na druhém místě nástroj XEN a na třetím nástroj VirtualBox. Použití vláken v tomto testu mělo pozitivní vliv na rychlost algoritmu. Jestliže byl zvyšován počet vláken, klesala doba, za kterou byl test ukončen.

Seznam použité literatury

[1] BERAN, Radek. Virtualizace operačních systémů. [online]. 2006, 30.11.2006 [cit. 2012-05-09]. Dostupné z: http://www.beranr.webzdarma.cz/virtualizace.html

[2] MATYSKA, Luděk. Techniky virtualizace počítačů (2). *Zpravodaj ÚVT MU* [online]. XVII, č. 3 [cit. 2012-05-09]. ISSN 1212-0901. Dostupné z: http://www.ics.muni.cz/zpravodaj/articles/545.html

[3] PATKA, Lukáš. Virtualizační technologie. Brno, 2009. Dostupné z: http://is.muni.cz/th/72735/fi_m/DP_PatkaL.txt. Diplomová práce. Masarykova univerzita. Vedoucí práce Ing. Mgr. Zdeněk Říha, Ph.D.

[4] MILATA, Martin. Pokročilé architektury počítačů: Virtualizace. [online]. [cit. 2012-05-08]. Dostupné z: http://wh.cs.vsb.cz/mil051/images/b/bb/PAP-PR-05_Virtualizace_a_jej%C3%AD_hardwarov%C3%A9_podpora.pdf

[5] Translation Lookaside Buffer. [online]. 2005, 13. 4. 2005 [cit. 2012-05-08]. Dostupné z: http://www.cs.nmsu.edu/~pfeiffer/classes/473/notes/tlb.html

[6] CHMIEL, PH.D., Ing. Pavel. Komunikace procesoru s okolím: Řadič přerušení IRQ. 2010. Dostupné z: http://chmiel.chytry.cz/files/ovt_epo_ps/et/cast1_11_podpurne_ obvody_ cpu.pdf

[7] GANIER, CJ. What is Direct Memory Access (DMA)?. [online]. 2005, 21. 8. 2005 [cit. 2012-05-08]. Dostupné z: http://cnx.org/content/m11867/latest/

[8] MILICHOVSKÝ, Petr a Jakub PONIKELSKÝ. Projekt: Porovnání rychlosti algoritmu ve vybraných programovacích jazycích. Liberec, 2010. Bakalářský projekt. Technická univerzita v Liberci. Vedoucí práce doc. Ing. Jiřina Královcová, Ph.D.

[9] SUCHÝ, Miroslav. Úvod do virtualizace pomocí XENu. *Linux* [online]. 2007, 8. 3. 2007 [cit. 2012-05-08]. Dostupné z: http://www.root.cz/clanky/uvod-do-virtualizace-pomoci-xenu/

[10] CITRIX SYSTEMS, Inc. XEN [online]. 2005. vyd. 2005 [cit. 2012-05-08]. Dostupné z: http://bits.xensource.com/Xen/docs/user.pdf

[11] VMWARE, Inc. *VMware Documentation* [online]. 2012 [cit. 2012-05-08]. Citované stránky: Home, Documentation. Dostupné z: http://www.vmware.com/support/pubs/

[12] PHORONIX MEDIA. *Phoronix Test Suite* [online]. 2008-2012 [cit. 2012-05-08]. Citované stránky: Home, Documentation. Dostupné z: http://www.phoronix-test-suite.com/?k=home

[13] C, Kevin. Hardware Testing and Benchmarking Methodology. [online]. 2006 [cit. 2012-05-08]. Dostupné z: http://donutey.com/hardwaretesting.php

[14] PETERKA, Jiří. Virtual memory. [online]. 2011 [cit. 2012-05-08]. Dostupné z: http://www.earchiv.cz/a93/a311c120.php3

[15]VirtualBox. In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2012 [cit. 2012-05-09]. Dostupné z: http://cs.wikipedia.org/wiki/VirtualBox

Příloha A – Obsah CD

Bakalářská práce ve formátu PDF

Kompletní výsledky všech testů ve formátu PDF

Vlastní testovací nástroj

Příloha B – Naměřené hodnoty

Tab. 5. Kompletní naměřené výsledky testů

Windows		
Nástroj	VirtualBox	VMware
pbzip [s]	77,22	75,84
	76,94	75,51
	77,13	75,65
	77,34	75,12
	76,8	77,02
	77,07	74,78
	77,42	76,87
	76,34	74,13
	77,02	77,43
	76,55	74,59
Medián	76,84	75,58
lzma [s]	340,05	323,29
	338,85	321,65
	339,53	323,51
	338,74	322,74
	337,13	321,73
	340,76	323,99
	340,31	320,61
	338,98	322,21
	338,56	322,41
	337,91	322,27
Medián	338,915	322,34
7zip [MIPS]	1447	1663
	1446	1705
	1457	1701
	1446	1684
	1449	1691
	1451	1687
	1448	1700
	1453	1643
	1455	1698
	1450	1680
Medián	1449,5	1689

CentOS				
Nástroj	VirtualBox	VMware	Xen	Normal
pbzip [s]	80,6	68,64	71,77	37,08
	80,66	69,7	75,47	36,41
	80,45	68,87	74,76	35,81
	80,23	68,23	71,91	36,58
	79,97	69,26	73,12	36,63
	79,43	67,54	74,65	37,29
	79,8	69,12	72,85	37,06
	80,32	69,9	74,05	36,43
	78,75	68,58	75,08	36,87
	79,85	69,51	75,05	37,73
Medián	80,1	68,995	74,35	36,75
lzma [s]	316,61	316,87	313	289,56
	316,04	317,22	313,52	289,13
	315,84	314,65	312,54	288,54
	316,73	314,87	312,76	289,51
	314,94	314,12	312,74	288,84
	316,32	314,21	312,83	287,09
	315,82	314,75	313,01	289,16
	315,92	315,32	313,07	288,69
	316,14	315,57	312,92	289,29
	315,74	314,31	312,94	290,24
Medián	315,98	314,81	312,93	289,15
7zip [MIPS]	1370	1730	1777	2805
	1375	1730	1801	2890
	1364	1729	1812	2900
	1381	1731	1807	2894
	1377	1734	1789	2879
	1359	1727	1794	2890
	1382	1722	1800	2940
	1374	1729	1780	2915
	1377	1725	1785	2869
	1368	1736	1784	2855
Medián	1374,5	1729,5	1791,5	2890

Windows		
Nástroj	VirtualBox	VMware
gzip [s]	48,15	45,08
	62,16	44,31
	61,53	46,86
	55,15	43,19
	62,9	45,5
	63,06	45,49
	59,67	45,86
	74,13	47,52
	70,32	48,77
	75,99	47,13
Medián	62,53	45,68
himeno	465,71	498,72
[MFLOPS]	465,43	492,77
	474,06	487,75
	466,88	491,92
	467	490,84
	468,34	489,42
	469,21	491,13
	473,96	495,15
	472,88	493,92
	472,89	487,54
Medián	468,775	491,525
ramspeed	2510,23	2700,39
[MB/s]	2460,08	2987,83
	2454,25	2760,32
Průměr	2474,853	2816,18
stream-copy	3150,41	2874,41
[MB/s]	3068,38	2875,92
	3115,08	2876,37
Průměr	3111,29	2875,56

CentOS				
Nástroj	VirtualBox	VMware	Xen	Normal
gzip [s]	63	39,95	48,39	21,58
	60,66	38,38	48,45	21,45
	60,6	38,83	48,22	20,27
	62,51	37,91	47,76	21,76
	59,97	38,65	46,9	20,07
	60,16	40,06	47,23	22,16
	57,8	38,69	48,29	21,38
	65,34	39	47,87	21,4
	62,17	38,6	48,3	21,51
	60,95	38,74	48,64	21,8
Medián	60,805	38,715	48,255	21,48
himeno	493,48	509,61	503,43	540,57
[MFLOPS]	489,84	513,86	501,2	541,78
	486,77	521,95	506,6	527,29
	486,21	500,53	507,5	534,93
	470,41	513,67	501,17	532,21
	478,95	520,72	503,09	530,15
	486,05	518,61	508,21	531,8
	491,56	521,24	500,96	519,71
	490,23	527,02	503,34	527,86
	470,3	522,91	502,6	524,03
Medián	486,49	519,665	503,215	530,975
ramspeed	2559,69	2737,64	3053,52	2850,29
[MB/s]	2563,3	2734,08	2833,66	2396,17
	2579,98	2694,57	2892,5	2397,02
Průměr	2567,656	2722,096	2926,56	2547,82
stream-copy	5442,65	3057,53	3199,5	3047,65
[MB/s]	5372,48	3053,34	3205,5	3067,68
	5354,42	3054,29	3207,8	3060,19
Průměr	5389,85	3055,05	3204,3	3058,50

Windows		
Nástroj	VirtualBox	VMware
stream-scale	3006,22	2811,96
[MB/s]	3017,08	2805,79
	3017,08	2801,04
Průměr	3013,46	2806,26
stream-add	3398,63	3185,07
[MB/s]	3400,88	3184,88
	3289,2	3185,69
Průměr	3362,90	3185,21
aiostress	21,38	42,58
[MB/s]	23,54	38,05
	22,43	38,45
	21,6	40,65
	22,47	41,26
	21,76	39,76
	21,51	39,52
	22,45	28,48
Medián	22,095	39,64
unpack-linux	38,03	39,06
[s]	39,21	44,44
	39,65	42,2
	40,06	39,15
	39,16	38,68
	41,84	47,91
	41,18	41,96
	41,74	43,65
Medián	39,855	42,08
apache	586,17	2719,04
[rps]	556,89	2716,52
	520,83	2720,41
	590,16	2719,32
	549,09	2750,21
	542,78	2756
	539,53	2755,75
	514,98	2753,55
Medián	545,935	2735,31

CentOS				
Nástroj	VirtualBox	VMware	Xen	Normal
stream-scale	5353,91	2981,92	3147,16	2995,34
[MB/s]	5395,48	2983,5	3147,85	2986,39
	5380,21	2980,89	3147,33	2995,87
Průměr	5376,53	2982,10	3147,44	2992,53
stream-add	6625,66	3573,37	3807,15	3515,09
[MB/s]	6536,42	3596,32	3803,19	3520,88
	6637,05	3590,44	3808,6	3512,23
Průměr	6599,71	3586,71	3806,31	3516,06
aiostress	8,33	24,18	6,79	32,61
[MB/s]	8,89	27,22	7,37	45,41
	9,52	25,86	6,42	33,75
	10,42	26,15	7,82	38,6
	11,96	27,83	6,92	39,4
	12,06	29,62	7,01	43,72
	10,22	25,14	6,64	34,6
	11,86	28,65	7,21	38,61
Medián	10,32	26,685	6,965	38,605
unpack-linux	36,53	37,63	28,71	22,53
[s]	33,55	36,19	26,76	20,39
	35,5	35,51	25,41	20,51
	36,81	35,79	24,6	21,06
	32,56	33,54	24,94	22,53
	34,74	37,1	27,48	21,23
	34,12	33,39	27,25	22,36
	35,68	34,15	26,89	20,26
Medián	35,12	35,65	26,825	21,145
apache	579,49	2849,07	4237,17	5799,13
[rps]	580,62	2845,51	4265,41	6000,31
	500,59	2898,14	4187,52	5976,34
	556,46	2871,76	4239,42	5834,09
	559,31	2835,71	4164,5	5774,32
	510,53	2859,58	4280,1	5612,5
	540,5	2848,11	4245,6	5788,22
	512,71	2855,79	4243,41	5740,1
Medián	548,48	2852,43	4241,415	5793,675

Windows		
Nástroj	VirtualBox	VMware
gtkcombobox	95,23	116,38
[s]	94,85	116,37
	94,72	115,78
	90,72	119,33
	90,71	119,02
	95,34	119,18
	101,94	118,12
	102,93	118,31
	95,26	120,88
	90,81	117,23
Medián	95,04	118,215
gtkbutton	33,34	45,2
[s]	33,37	45,11
	33,3	45,1
	33,36	45,11
	33,32	45,06
	33,34	45,18
	33,56	45,03
	33,28	45,2
	33,33	45,21
	33,39	45,16
Medián	33,34	45,135
gtkpixbuffs	2,55	3,02
[s]	2,64	2,85
	2,58	2,85
	2,51	2,86
	2,47	2,85
	2,84	2,85
	2,49	2,86
	2,48	2,86
	2,47	2,88
	2,48	2,93
Medián	2,5	2,86

CentOS				
Nástroj	VirtualBox	VMware	Xen	Normal
gtkcombobox	109,23	125,44	89,52	45,42
[s]	112,35	127,6	88,68	45,93
	116,25	121,93	89,98	49,2
	113,38	132,99	88,01	51,28
	117,83	126,02	93,32	52,18
	117,76	129,73	97,09	53,09
	114,2	123,75	99,41	47,99
	116,12	129,11	100,37	51,82
	118,23	132,52	103,67	54,59
	115,5	128,57	96,98	55,2
Medián	115,81	128,085	95,15	51,55
gtkbutton	37,94	43,97	31,81	16,05
[s]	48,41	49,72	32,17	16,21
	40,52	49,98	33,36	16,48
	40,62	50,11	32,26	16,87
	37,48	50,61	33,57	17
	38,63	50,26	33,88	19,22
	37,53	50,48	33,2	18,33
	38,04	46,59	34,12	18,96
	37,48	50,15	34,21	20,88
	37,35	47,17	33,74	18,57
Medián	37,99	50,045	33,465	17,665
gtkpixbuffs	2,75	2,9	3,31	2,05
[s]	2,73	2,71	3,44	2,03
	3,07	2,7	3,52	2,03
	2,96	3,01	3,04	2,04
	2,72	2,71	3,36	2,03
	2,91	2,81	3,42	2,03
	2,64	2,73	3,58	2,03
	2,54	2,72	3,4	2,03
	2,51	3,17	3,58	2,04
	2,55	2,87	3,4	2,03
Medián	2,725	2,77	3,41	2,03

gtkdrawningarea	31,83	59,58
[s]	26,56	59,73
	47,6	59,93
	45,71	59,01
	44,61	60,43
	46,75	60,32
	48,66	59,92
	47,02	57,75
	47,37	60,2
	47,42	58,82
Medián	46,885	59,825

gtkdrawningarea	67,2	63,31	60,07	32,49
[s]	78,54	73,67	56,99	33,2
	85,4	70,74	61,16	38,09
	89,08	66,14	58,42	34,52
	88,71	68,16	54,07	39,03
	85,51	63,17	48,1	35,11
	85,87	68,06	59,25	32,88
	80,51	70,35	57,6	33,48
	79,5	68,76	57,69	38,1
	83,4	73,73	58,28	36,85
Medián	84,4	68,46	57,985	34,815

Windows		
Nástroj	VirtualBox	VMware
J2d	131367,95	117053,41
[ups]	131337,6	117271,55
	131510,75	117229
	131667,69	117021,44
	131669,79	116690,89
Průměr	131510,76	117053,25

CentOS				
Nástroj	VirtualBox	VMware	Xen	Normal
J2d	136825,39	119464	143605,17	166571,41
[ups]	139842,99	124541,07	143759,63	167135
	139102,83	122698,37	143968,4	166395,36
	129757,5	122455,58	143516,85	165667,22
	138598,2	124332,82	143948,09	166207,44
Průměr	136825,38	122698,36	143759,62	166395,28

glxgears	637	924
[FPS]	637	993
	636	1000
	636	1002
	632	999
	636	970
	638	980
	632	987
	633	994
	630	998
	636	993
	633	993
	630	992
	632	980
	636	987
Medián	636	993

glxgears	791	764	1366	1469
[FPS]	767	814	1364	1461
	728	818	1365	1460
	785	790	1366	1460
	785	817	1366	1461
	775	813	1367	1461
	787	799	1367	1461
	811	821	1367	1460
	812	817	1365	1461
	817	801	1367	1462
	812	797	1367	1482
	817	767	1366	1439
	810	782	1364	1460
	799	746	1365	1461
	812	772	1367	1461
Medián	799	799	1366	1461